МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Чорноморський національний університет
імені Петра Могили
Факультет комп'ютерних наук
Кафедра інтелектуальних інформаційних систем

ДОПУЩЕНО ДО ЗАХИСТУ

Завідувач кафедри інтелектуальних

інформаційних систем, д-р техн. наук, проф.
_____ Ю.П. Кондратенко

«___»_____2023 року

БАКАЛАВРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

ОСВІТНЯ СОЦІАЛЬНА МЕРЕЖА ДЛЯ ВИКЛАДАЧІВ ТА СТУДЕНТІВ

Спеціальність 122 «Комп'ютерні науки»

122-БКР-401.21910116

Виконав студент 4-го курсу, групи 401
__*_____*В.О. Павлюх

«___» червня 2023 р.

Керівник: д-р. фіз.-мат. наук, професор

____*_____*Е.А. Лисенков

«___» червня 2023 р.

Миколаїв – 2023

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Чорноморський національний університет
імені Петра Могили
Факультет комп'ютерних наук
Кафедра інтелектуальних інформаційних систем

Рівень вищої освіти **бакалавр**

Спеціальність **122 «Комп'ютерні науки»**

(шифр і назва)

Галузь знань **12 «Інформаційні технології»**

(шифр і назва)

ЗАТВЕРДЖУЮ

Завідувач кафедри інтелектуальних інформаційних систем, д-р техн. наук, проф.

_____Ю. П. Кондратенко

«_____» 20 р.

З А В Д А Н Н Я

на виконання кваліфікаційної роботи

Видано студенту групи 401 факультету комп'ютерних наук Павлюху Владиславу Олександровичу.

1. Тема кваліфікаційної роботи «Освітня соціальна мережа для викладачів та студентів».

Керівник роботи Лисенков Едуард Анатолійович, доктор фіз.-мат. наук, професор.

Затв. наказом Ректора ЧНУ ім. Петра Могили від «___» _____202_ р. №

2. Строк представлення кваліфікаційної роботи студентом «___»_____ 202_ р.

3. Вхідні (початкові) дані до роботи: діяльність освітніх соціальних мереж для викладачів та студентів, база даних початкових користувачів.

Очікуваний результат: Повністю функціонуюча освітня соціальна мережа.

4. Перелік питань, що підлягають розробці (зміст пояснювальної записки):

– аналіз актуальності соціальних систем у сучасному світі, стан освіти в Україні під час очного та заочного навчань;

– обґрунтування вибору технологій та засобів розробки системи;

– проектування освітньої соціальної мережі;

– розробка та здійснення програмної реалізації освітньої соціальної мережі для викладачів та студенті.

5. Перелік графічного матеріалу: презентація, -- рисунків, --додатків.

6. Завдання до спеціальної частини: «Оцінка ризиків пов'язаних з роботою на ПК та заходи їх зниження».

7. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис
Спеціальна частина з охорони праці	к.т.н, доцент кафедри екології Алексєєва А. О.	

Керівник роботи_____доктор фіз.-мат. наук, проф. Лисенков Е. А.____

(наук. ступінь, вчене звання, прізвище та ініціали)

_____

(підпис)

Завдання прийнято до виконання___Павлюх В. О._____

(прізвище та ініціали)

_____

(підпис)

Дата видачі завдання « »_____2022р.

КАЛЕНДАРНИЙ ПЛАН
виконання кваліфікаційної роботи

Тема: «Освітня соціальна мережа для викладачів та студентів»

№	Найменування роботи	Початок	Закінчення	Примітки
1	Подання заяви на затвердження теми та керівника БКР	26.10.2022	30.10.2022	Виконано
2	Отримання завдання на виконання БКР	25.11.2022	25.11.2022	Виконано
3	Складання календарного плану роботи на весь період виконання БКР	26.11.2022	10.12.2022	Виконано
4	Аналіз предметної області, існуючих аналогів у моніторингу та аналізу діяльності користувача ПК	11.12.2022	31.12.2022	Виконано
5	Створення дизайну, проєктування та програмна реалізація застосунку	01.04.2023	01.05.2023	Виконано
6	Отримання завдання на переддипломну практику	02.05.2023	02.05.2023	Виконано
7	Проходження переддипломної практики	02.05.2023	14.05.2023	Виконано
8	Збір та аналіз матеріалів, оформлення розділів БКР	01.05.2023	14.05.2023	Виконано
9	Робота над розділами пояснювальної записки БКР	14.05.2023	27.05.2023	Виконано
10	Розробка спеціальної частини з охорони праці	24.05.2023	29.05.2023	Виконано
11	Попередній захист БКР	29.05.2023	29.05.2023	
12	Доробка та остаточне оформлення БКР	29.05.2023	19.06.2023	
13				

Розробив студент Павлюх В. О. _____

(прізвище та ініціали) *(підпис)*

Керівник роботи доктор фіз-мат. наук, професор Лисенков Е.А.

_____

(наук. ступінь, вчене звання, прізвище та ініціали) *(підпис)*

« » 202_ р.

АНОТАЦІЯ
бакалаврської кваліфікаційної роботи студента групи 401 ЧНУ ім. Петра Могили

Павлюха Владислава Анатолійовича

Тема: «Освітня соціальна мережа для викладачів та студентів»

Бакалаврська кваліфікаційна робота присвячена проектуванню, розробці, програмній реалізації та впровадженню освітньої соціальної мережі для викладачів та студентів.

Об'єкт дослідження – функціонування освітньої платформи з функціоналом соціальної мережі.

Предмет дослідження – web-орієнтовані програмні засоби для освітньої соціальної мережі для викладачів та студентів.

Метою роботи є підвищення ефективності роботи освітньої платформи для студентів та викладачів, шляхом використання удосконалених чат-технологій.

Бакалаврська кваліфікаційна робота робота складається з фахової частини і спеціальної частини з охорони праці. Пояснювальна записка дипломної роботи складається зі вступу, чотирьох розділів, висновків та додатків.

У першому розділі здійснено аналіз рівня сучасної освіти в Україні, її стан та недоліки. Також доведено актуальність створення освітніх соціальних мереж. У другому розділі розглядаються технології, методи та моделі проектування розробки системи. У третьому розділі описано вхідні дані та проектування інформаційної системи. У четвертому розділі описано програмну реалізацію освітньої соціальної мережі.

Дипломна робота містить 102 сторінки, 57 рисунків, 27 джерел, 6 додатків.

ABSTRACT

for bachelor's qualification work of a student of group 401 of Petro Mohyla Black Sea National University

Vladyslav's Pavliukh

Subject: «Educational social network for teachers and students»

Thesis is devoted to the design, development, software implementation and implementation of the information system of an educational social network for teachers and students.

The object of research is the educational platform with social network functionality.

The subject of the research is a Web-based software for an educational social network for teachers and students.

The purpose of the thesis is to create such a communication system of students and teachers, which will combine the functionality of Telegram, for instance, and Moodle. So that the two sides could work conveniently and complete everything within the specified time.

Thesis consists of a professional part and a special part on labor protection. Explanatory note of the thesis consists of an introduction, four chapters, conclusions and appendix.

The first section analyzes the level of modern education in Ukraine, its condition and negative sides. The relevance of creating educational social networks has also been proven. In the second section deals with technologies, methods and models of system's development design. The third section describes the input data and information system design The fourth chapter describes the software implementation of the educational social network.

Thesis contains 102 pages, 57 figures, 27 sources, 6 supplements.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

БД – база даних

ГН – гігієнічні норми

ЕПТ – Електрично – променева трубка

КНО – Коефіцієнт природної освітленості

КТ – комп’ютерна техніка

ОС – операційна система

ПЗ – програмне забезпечення

AOT – Ahead-of-Time

CSS – Cascading Style Sheets

DLL – Dynamic Link Library

FDD – Feature Driven Development

HTML – Hyper Text Markup Language

MVC – Model View Control

ORM – Object-Relational Mapping

ВСТУП

Актуальність Створення власної освітньої соціальної мережі є як ніколи актуальним. Карантин за часів COVID-19 показав всьому людству потребу бути готовим до дистанційного навчання та роботи. У випадку студентів нас рятувала онлайн платформа MOODLE, налаштована під наш університет. Але під час повномасштабної війни навіть така система зазнає значної шкоди через відключення світла, через що навчальний процес ускладнюється в рази.

Метою роботи є підвищення ефективності роботи освітньої платформи для студентів та викладачів, шляхом використання удосконалених чат-технологій.

Відповідно до поставленої мети було сформульовано **завдання дослідження:**

– здійснити аналіз сучасного стану дистанційної освіти в Україні. Переглянути прототипні соціальні мережі, розібрати їх плюси та мінуси;

– затвердити стек технологій та засобів розробки обраної освітньої соціальної мережі;

– розробити та здійснити програмну реалізацію освітньої соціальної мережі викладачів та студентів.

Об’єкт дослідження – функціонування освітньої платформи з функціоналом соціальної мережі.

Предмет дослідження – web-орієнтовані програмні засоби для освітньої соціальної мережі для викладачів та студентів.

Методологічною основою дослідження є аналітичні та соціологічні методи, які дозволили вивчити предмет та об’єкт дослідження, дослідити напрями та шляхи реалізації зручної мережі.

Практичне значення отриманих результатів полягає в тому, що використання розробленої освітньої мережі дозволить вирішити низку проблем при потенційних відключеннях світла у майбутньому.

1 АНАЛІЗ ПРЕДМЕТНОЇ СФЕРИ. ПОСТАНОВКА ЗАДАЧІ

1.1 Опис предметної сфери

В умовах наростання проблем які несуть глобальну загрозу можливості сталого розвитку деяким країнам, освіта та наука бере на себе завдання виховати фахівців, які зможуть впоратися з цими проблемами. Наука виконує тепер місію збереження балансу між природою та людиною. Тепер головною місією освіти стає не лише передача знань, але й розкриття творчого потенціалу кожної людини. Для досягнення цієї місії необхідна координація зусиль вчених з різних галузей, активної участі громадського сектору. Економічний, політичний, соціальний розвиток України буде залежати від ефективного поєднання креативу, творчості та професіоналізму кожного українського громадянина.

Як відомо, економіка залежить від можливостей освіти, тому що освіта формує певний рівень знань, вмінь та навичок. Тобто, якщо немає розвитку освіти, то і не є можливим розвиток економіки держави, і це потрібно чітко усвідомлювати. Саме тому країни які вкладали ресурси в якісну освіту, яка дозволяє формувати професіоналів, та дає можливість всебічного розвитку людині, змогли досягти успіхів в економіці.

У всі часи освіта була фактором розвитку та прогресу, вона дозволяла розвиватися культурі та економіці, розкривати потенціал людини. Сьогодні вектор розвитку освіти визначається такими нормативно-правовими актами як національна програма «Освіта», ЗУ «Про вищу освіту», 28 Указом Президента України «Про основні напрями реформування вищої освіти в Україні».

Політика Української держави спрямована на розвиток освіти та досягнення нею рівня прогресивних країн світу. Це включає в себе осучаснення змісту, програм, розробка нових більш ефективніших методів навчання.

«Розбудова системи освіти, її докорінне реформування, – наголошується в національній програмі «Освіта» (Україна XXI століття), – мають стати основою відтворення інтелектуального, духовного потенціалу народу, виходу вітчизняної науки, техніки і культури на світовий рівень, національне відродження, становлення державності та демократизації суспільства в Україні».

Зміни які відбулися в Україні, не могли не позначитися на освіті. Якщо в СРСР всі освітні заклади були державні, то після його розпаду Україна як самостійна і незалежна держави яка переходила від планової до ринкової економіки почала створювати усі умови для формування приватних ВНЗ. Це зумовило розширення освітніх послуг та сприяло створенню конкурентного середовища на ринку освітніх послуг [1].

На сьогодні українська освіта потребує реформування у таких напрямках як управління цією системою, прийняття сприятливих для розвитку освіти нормативно-правових норм, розробка програм підтримки стратегічних напрямків освіти, зокрема пільгових кредитів, та більш гнучкої системи оподаткування, інтеграції до світового наукового товариства [2, 3].

Варто зауважити, що Україна отримала у спадок від СРСР досить непогану систему вищої школи. Наприклад суттєво збільшилась кількість ВНЗ в Україні в період 1991-2007 рр, з 149 до 350. Серед нині діючих вищих навчальних закладів III–IV рівня акредитації 236 знаходяться у державній та комунальній власності, а 114 – приватні. Усього в Україні навчається 2 млн. 635 тис. 395 осіб, з них за бюджетною формою навчання – 1 млн. 74 тис. 280 студентів [4, 5].

Нині в Україні понад 560 студентів припадає на 10 тис. населення. Це відповідає середньоєвропейським показникам і є передумовою входження нашої держави до кола цивілізованих 29 країн з розвиненою, соціально орієнтованою, регульованою ринковою економікою [6, 7].

Швидкий розвиток інформаційних технологій не зміг обійти стороною освітній процес. У сучасній навчальному процесі відбулися зміни поняття навчання: засвоєння знань поступилось місцем умінню знаходити з різноманітних джерел інформацію та користуватися нею. Слід усвідомлювати, що нині відбувається інтеграція освітнього середовища у глобальний інформаційний простір, і якщо будуть упущені певні кроки в даному напрямку, то такий важливий інститут, як освітні заклади, можуть залишитися поза рамками розвитку сучасного суспільства [8, 9].

Сучасне суспільство бажає бачити навчальні заклади не лише одним з освітніх ресурсів, а, скоріше, простором розвитку та співпраці як всередині, так і стосовно до зовнішнього світу. Сучасні освітні веб-портали є складовою частиною навчально-виховного процесу, мають навчально-методичне призначення і використовується для забезпечення навчальної діяльності учнів, студентів і вважаються одним із головних елементів інформаційно-освітнього середовища.

Сучасні світові науковці й практики трактують освітнє середовище, як частину життєвого, соціального середовища людини, яка виявляється у сукупності усіх освітніх факторів, що безпосередньо або опосередковано впливають на особистість у процесах навчання, виховання та розвитку; є певним виховним простором, в якому здійснюється розвиток особистості.

Під відкритим інформаційно-освітнім середовищем розуміють "єдиний інформаційно-освітній простір, який побудований за допомогою інтеграції інформації на традиційних та електронних носіях та містить електронні бібліотеки, розділені бази даних, оптимально структурований навчально-методичний комплекс та розширений апарат дидактики, в якому діють принципи нової педагогічної системи".

Без створення єдиного інформаційного освітнього простору навчального закладу неможливо розв’язати задачу формування особистості індивідууму, готової до життєдіяльності на інформативній основі, адже саме з його допомогою можна перейти на якісно новий рівень у підходах до використання комп’ютерної техніки й

інформаційних технологій у всіх структурних підрозділах школи, що буде сприяти підвищення якості навчання й ефективності управління школою.

Навчальний веб-портал є ефективним інструментом для вирішення задачі розширення освітніх можливостей заочного навчання, організації дистанційної освіти, відображення діяльності студентів і педагогів для зовнішніх відвідувачів мережі Інтернет, інформаційної підтримки студентів і викладачів, проведення дистанційних батьківських зборів, семінарів, конкурсів [10].

Дистанційна вища освіта в Україні являє собою освітні програми, що надаються онлайн або за допомогою відеоконференційного зв'язку. Такий формат освіти дозволяє студентам займатися навчанням у зручний для них час і з будь-якої точки світу [11, 12, 13].

До головних проблем дистанційної вищої освіти в Україні можна віднести:

– низьку якість навчання. Це може бути пов'язано з тим, що дистанційна освіта ще не отримала достатньої уваги від викладачів та інших спеціалістів;

– недостатню кількість професійних викладачів, які б змогли дати якісну освіту студентам. Багато з викладачів не мають достатньої підготовки для роботи з онлайн-освітою;

– також важко забезпечити рівні умови навчання для всіх студентів. Не всі студенти мають рівний доступ до зручних комп'ютерів, швидкого Інтернету та інших необхідних засобів для успішного навчання;

– відсутність можливості для студентів спілкуватися між собою та з викладачами у режимі реального часу. Це може призвести до втрати мотивації та важкостей у вирішенні навчальних завдань;

– важкість забезпечити ефективну систему контролю знань та оцінювання студентів у дистанційному форматі. Не всі методи тестування, які застосовуються у дистанційному навчанні, можуть забезпечити достатню об'єктивність та точність оцінки;

– проблемою є також відсутність можливості для студентів здійснювати практичну підготовку на певних спеціальностях. Багато професій потребують практичних занять та взаємодії з людьми в реальному житті, що не завжди можливо забезпечити у дистанційному форматі;

– крім того, у дистанційному навчанні важко забезпечити студентам належну підтримку від викладачів та інших фахівців. Зокрема, студенти можуть мати складнощі з розумінням матеріалів або виконанням завдань, тому вони потребують підтримки та консультацій з боку викладачів;

– проблема несумісності дистанційного навчання з іншими важливими аспектами життя, такими як робота або родина. Це може призвести до того, що студенти не можуть домогтися належного рівня успішності у навчанні.

На фоні повномасштабної війни, енергосистема України переживала найтяжчі часи. Шкільні платформи, що хостились на фізичних серверах в межах університетів, «підводили» і викладачів і студентів. Осінь з зимою видались тяжкими, частково це було пов’язано і з відключеннями світла. Пропоную поглянути динаміку запитів українців стосовно відключень на рис. 1.1 та рис. 1.2.

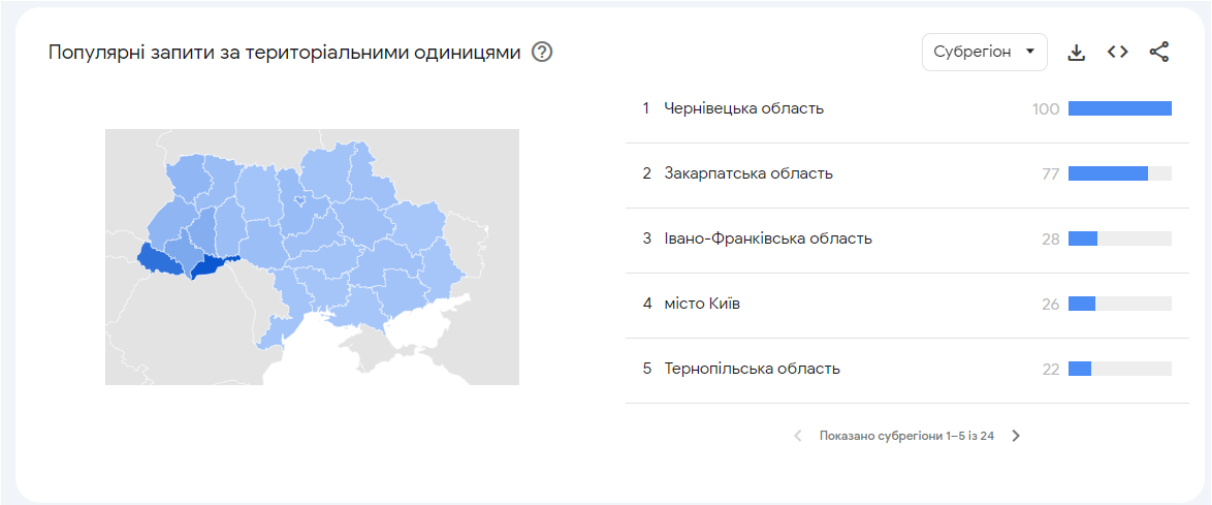


Рисунок 1.1 – Ситуація по запитам відключення по регіонам країни

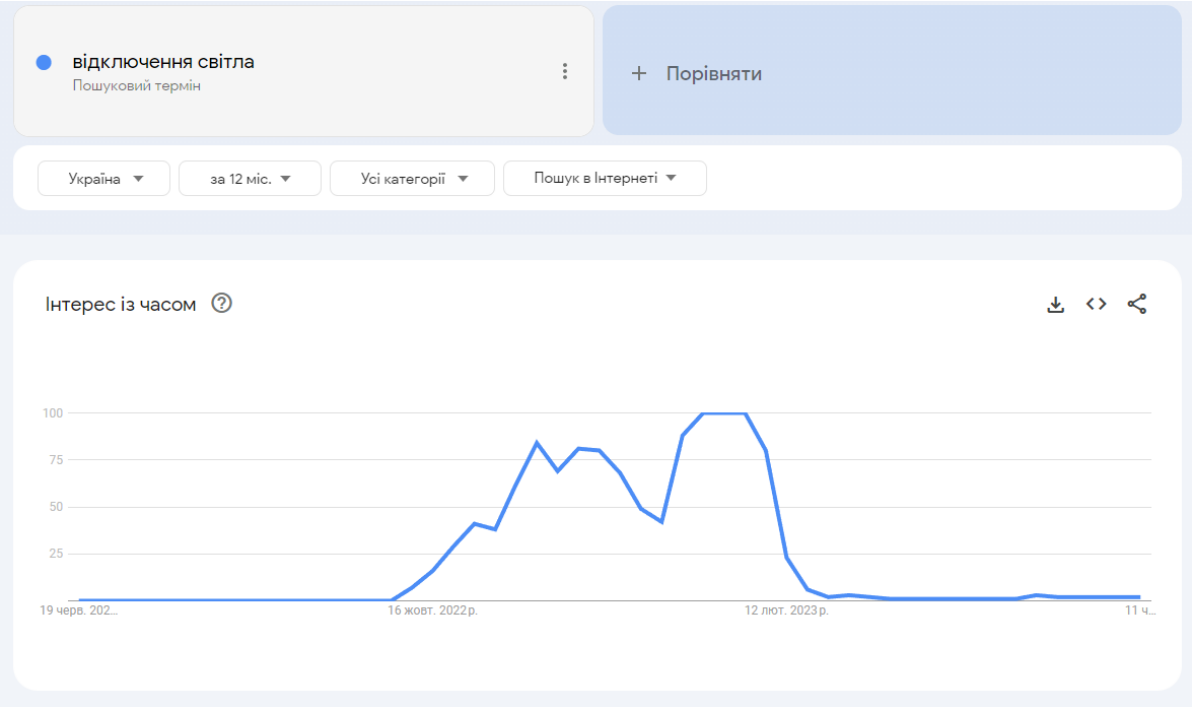


Рисунок 1.2 – Актуальність запитів про відключення світла в Україні за рік

Проаналізувавши ці дані можна зробити висновок, що актуальність виконання роботи саме у цій сфері є актуальним і потребує певного рішення.

1.2 Огляд та аналіз наявних аналогів та публікацій

Освітні соціальні мережі мають кілька загальних переваг, які сприяють спільному навчанню, співпраці та обміну знаннями. У світі вже існує декілька освітніх соцмереж, таких як Edmodo, Schoology, Moodle, Coursera тощо. Кожна з них має свої переваги та недоліки.

Edmodo. Це проста у використанні освітня соціальна мережа, спеціально створена для освітніх потреб. Вона надає можливість завантажувати інформацію, завдання та спілкуватися у захищеному середовищі. Користувачі мають зручний доступ до обміну ресурсами та обговорень. Інтерфейс зображено на рис. 1.3.

Однак, порівняно з загальними соціальними мережами, Edmodo має обмежений функціонал. Крім того, в порівнянні з іншими платформами, кількість користувачів Edmodo може бути меншою, що обмежує масштаб спілкування.

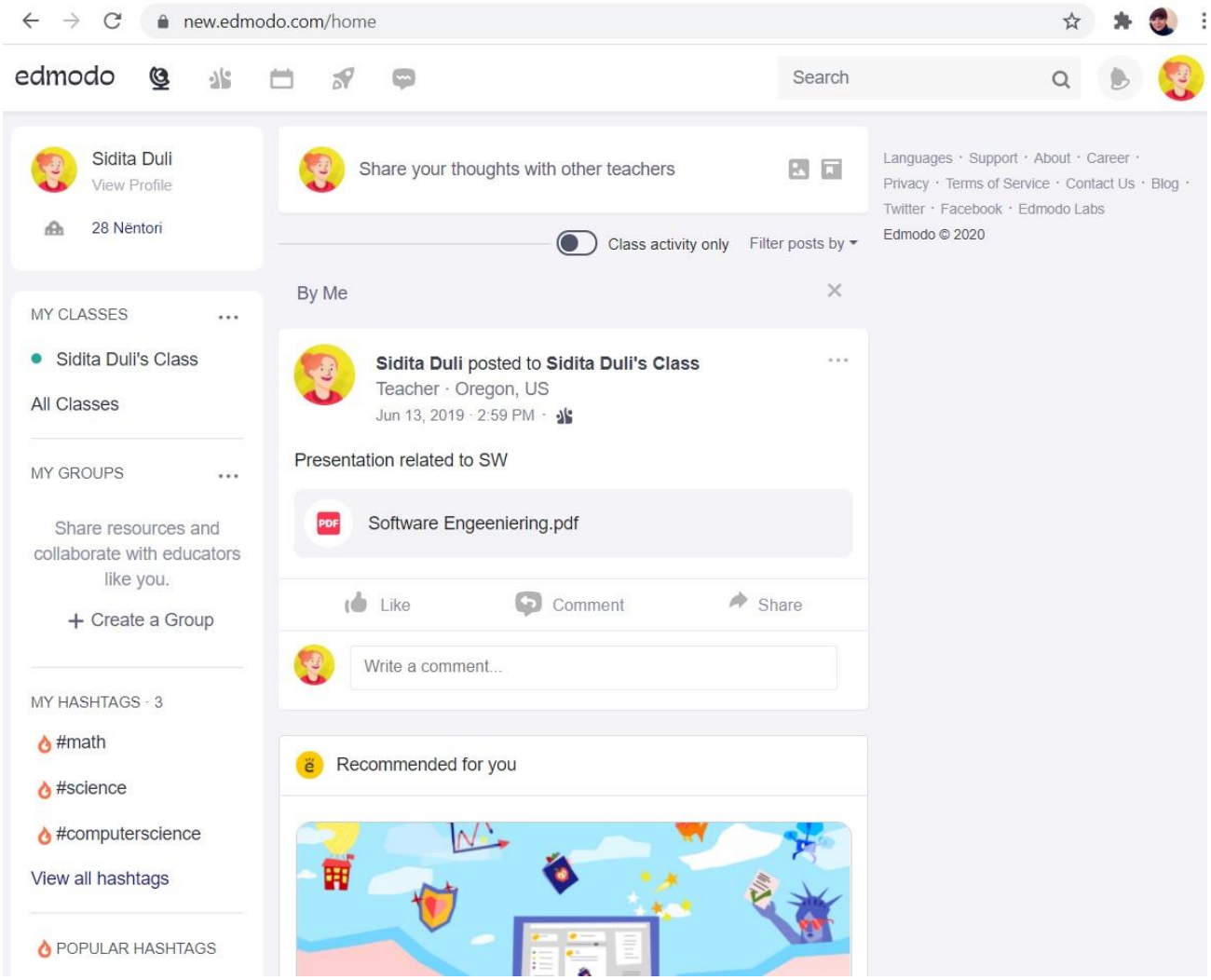


Рисунок 1.3 – Інтерфейс освітньої соціальної мережі Edmodo

Schoology пропонує широкий функціонал для освітніх потреб. Вона дозволяє зручно керувати курсами, відстежувати прогрес студентів, оцінювати їхні досягнення. Також Schoology надає зручні інструменти для співпраці та комунікації між вчителем та учнями. Інтерфейс зображено на рис. 1.4.

Однак, деяким новачкам інтерфейс Schoology може здаватися складним. Крім того, деякі функції, наприклад, додаткові інтеграції з іншими інструментами, доступні лише у платній версії.

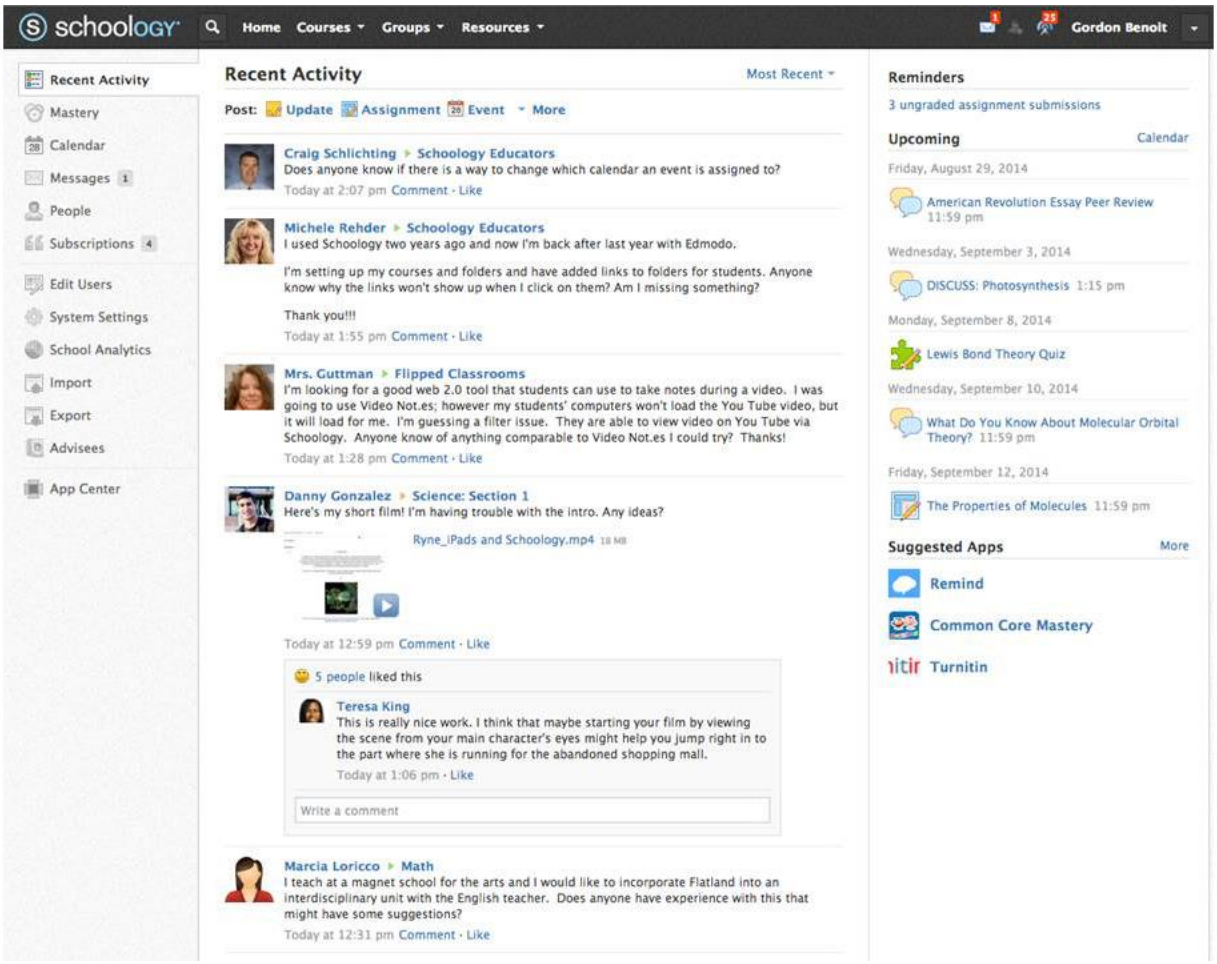


Рисунок 1.4 – Інтерфейс освітньої соціальної мережі Schoology

Moodle - популярна освітня соціальна мережа з великою спільнотою користувачів. Вона базується на відкритому програмному забезпеченні, що дає можливість налаштувати індивідуальні курси, завдання та ресурси. Moodle також надає широкі можливості для взаємодії та спілкування між користувачами. Інтерфейс зображено на рис. 1.5.

Однак, інтерфейс Moodle може бути менш привабливим і менш інтуїтивно

зрозумілим порівняно з деякими іншими соціальними мережами. Крім того, використання Moodle вимагає певних технічних навичок для налаштування та управління платформою.

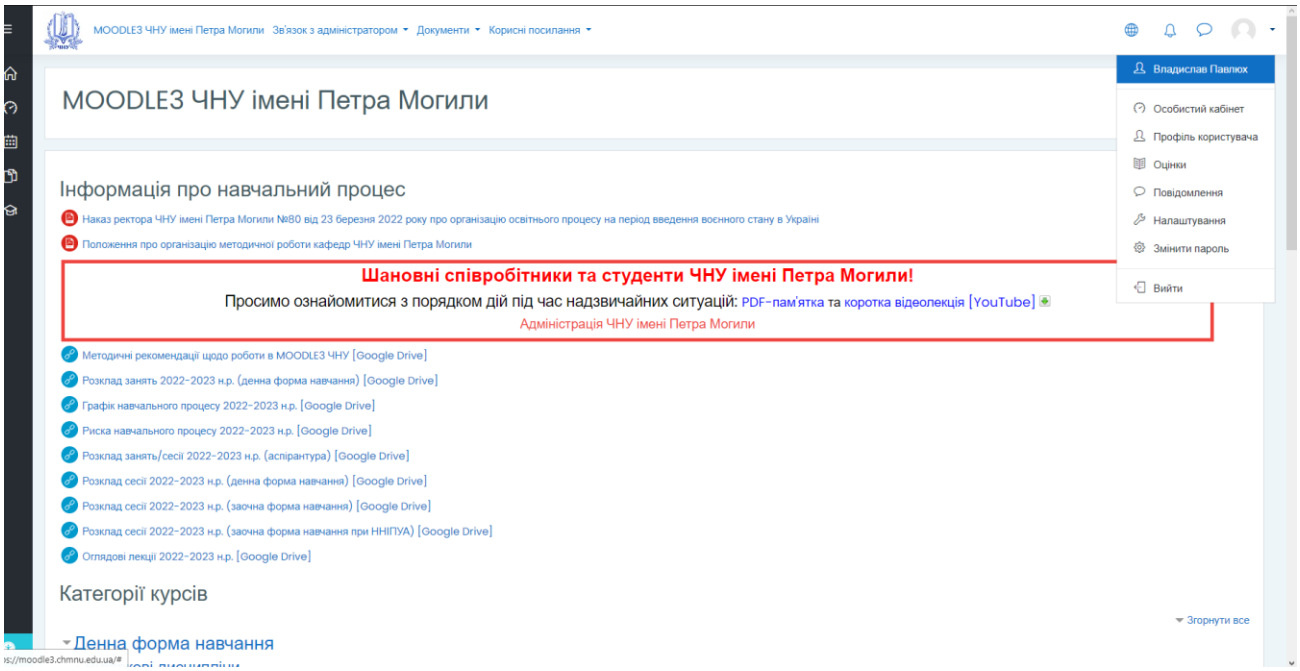


Рисунок 1.5 – Інтерфейс освітньої соціальної мережі Moodle

Classroom by Google пропонує зручну інтеграцію з іншими інструментами Google, такими як Google Документи та Google Календар. Платформа надає зручні можливості для спільної роботи над завданнями, збереженням та обміном документами. Інтерфейс Classroom by Google є дружнім як для вчителів, так і для студентів, зображено на рис. 1.6.

Однак, Classroom by Google може бути обмежений у функціоналі порівняно з деякими іншими освітніми соціальними мережами. Він не має такої широкої функціональності для взаємодії між користувачами, що може бути важливим для деяких освітніх сценаріїв.

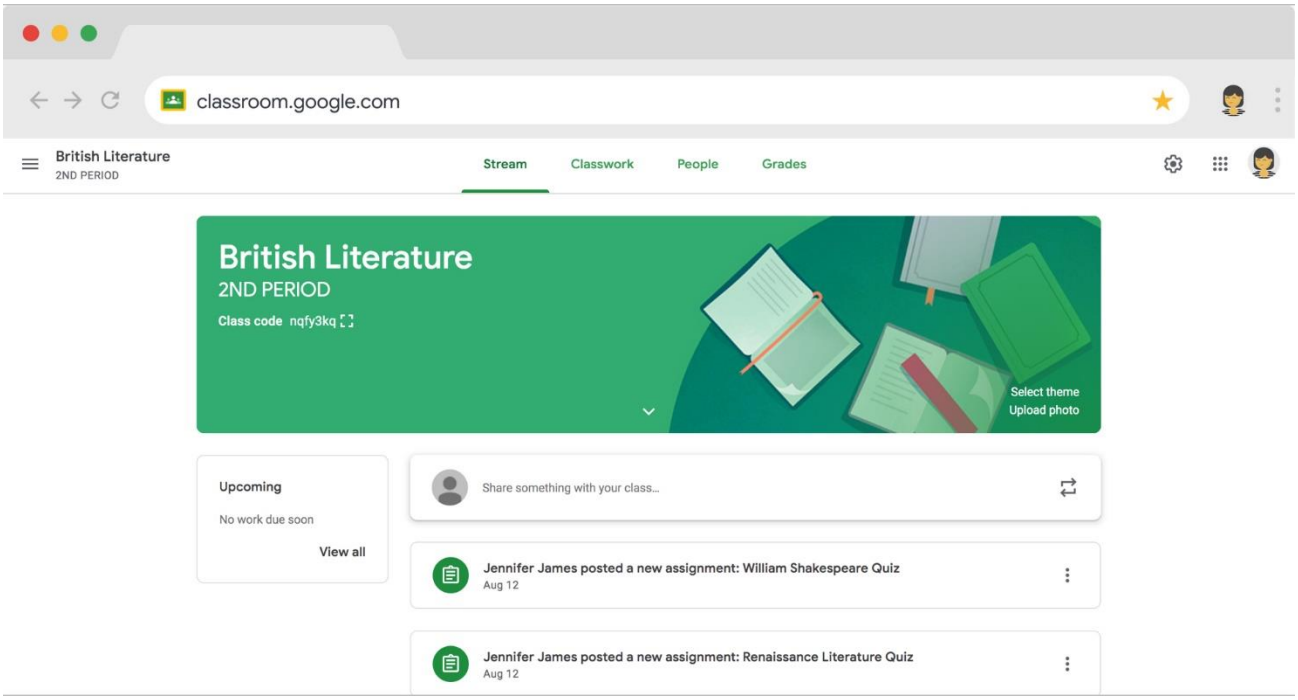


Рисунок 1.6 – Інтерфейс освітньої соціальної мережі Classroom

Підсумовуючи, маємо, що загальний аналіз існуючих освітніх соціальних мереж показує, що вони надають зручну платформу для спілкування, співпраці та обміну знаннями. Кожна з цих мереж має свої переваги, такі як спеціалізація на освітніх потребах (Edmodo), широкий функціонал для керування курсами (Schoology), відкрите програмне забезпечення та налаштування (Moodle) або інтеграцію з іншими інструментами Google (Classroom by Google). Однак, кожна з них також має свої обмеження, такі як обмежений функціонал, складний інтерфейс або обмежені можливості взаємодії.

1.3 Постановка задачі дослідження

Розробка соціальної мережі із поєднанням корисних цілей, таких як можливість проведення навчання, є актуальною проблемою розвитку мережевих сервісів підтримки його діяльності. Здійснений аналіз стану мережі на подібні платформи та мережі дозволив висунути наступні вимоги до нашої системи мережі, які описані нижче.

1. Серед користувачів доцільно виділити адміністраторів, викладачів та студентів.
2. Адміністратор має право на: усі можливі дії по функціоналу сайту. Це основна особа платформи, оскільки вона займається додаванням нових викладачів, має змогу разом з ними створювати нові предмети а також форми для домашніх завдань. До можливостей адміністратора також відноситься видалення студентів, що не відповідатимуть своїй групі.
3. Викладач має право на: створення нового розділу з предмету, додавання домашнього завдання до нього, а також оцінку прикріплених файлів від студентів.
4. Студент має право на: реєстрацію свого аккаунту, перегляд домашніх завдань, можливість завантажити файл з виконаним домашнім завданням відповідно до потрібного розділу.
5. Створення лаконічного та простого користувацького інтерфейсу.
6. Для роботи соціальної мережі повинна бути спроектована база даних, яка міститиме інформацію про усіх користувачів різних ролей, назви предметів та розділів а також завантажені студентами файли.

Висновки до розділу 1

Виходячи з аналізу можна зробити висновок, що сучасний стан та умови електричного постачання наразі покращився, але немає ніякої гарантії, що подібне не повториться у майбутньому. У таких умовах завжди існуватиме потреба у впровадження соціальних освітніх систем для викладачів та студентів. Наявність таких мереж сприяє підвищенню ефективності роботи викладачів навчального закладу та студентів, їхньої здачі та вчасності перевірки робіт.

Було остаточно доведено актуальність обраної теми дипломної роботи а також розглянуто об’єкт предметної сфери. Навчальні платформи можна умовно поділити на такі типи:

– наукових досліджень;
– довідкового характеру;
– дистанційної освіти;
– змагальних та інформаційних проектів;
– консультативні, освітні портали.

Визначено, що сторінки навчальних мереж мають в своїй основі однакові структурні компоненти, відмінність яких полягає тільки в їх наповнені, дизайні та місці розташування.

В таких компонентах розміщується необхідна інформація та програмні модулі. З набору модулів, в залежності від цільового призначення мережі, формується скелет його функціональності.

2 МОДЕЛІ, МЕТОДИ ТА ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ ДЛЯ ВИРІШЕННЯ ПОСТАВЛЕНОЇ ЗАДАЧІ

2.1 Методи для вирішення задачі

Освітні соціальні мережі є спеціалізованими платформами, які забезпечують взаємодію між учнями, вчителями, батьками та іншими учасниками освітнього процесу. Ці мережі можуть містити різноманітні функціональні можливості, спрямовані на полегшення обміну знаннями, спілкування та співпрацю. При написанні освітньої соціальної мережі можна використовувати різні моделі, методи та інформаційні технології [14].

Особливості соціального графа характеризується такими метриками, як: метрики взаємин, метрики зв'язків та сегментації [15]. Для вирішення завдань на соціальному графі використовуються спеціальні моделі, за допомогою яких можна замінити «реальні» графи. За допомогою соціальних графів вирішують такі завдання, як: ідентифікація користувачів; соціальний пошук; генерація рекомендацій з вибору «друзів», медіа-контенту, новин, тощо; виявлення «реальних» зв'язків або збір відкритої інформації для моделювання графа. Обробка даних соціальних графів пов'язана з низкою проблем, як наприклад відмінності соціальних мереж, закритість соціальних даних.

Під соціальною мережею на якісному рівні розуміється соціальна структура, яка складається з множини агентів (суб’єктів – індивідуальних чи колективних, наприклад, індивідів, сімей, груп організацій) і визначеній на цій множині відношень (сукупності зв’язків між агентами, наприклад, знайомства, дружби, співпраці, комунікації) [16, 17]. Формально соціальна мережа представляється як граф G(N, E), в якому N = {1,2,…,n} – скінченна множина вершин (агентів) та E – множина ребер, що відображає взаємодію агентів [18]. Візуально зображено на рис. 2.1.

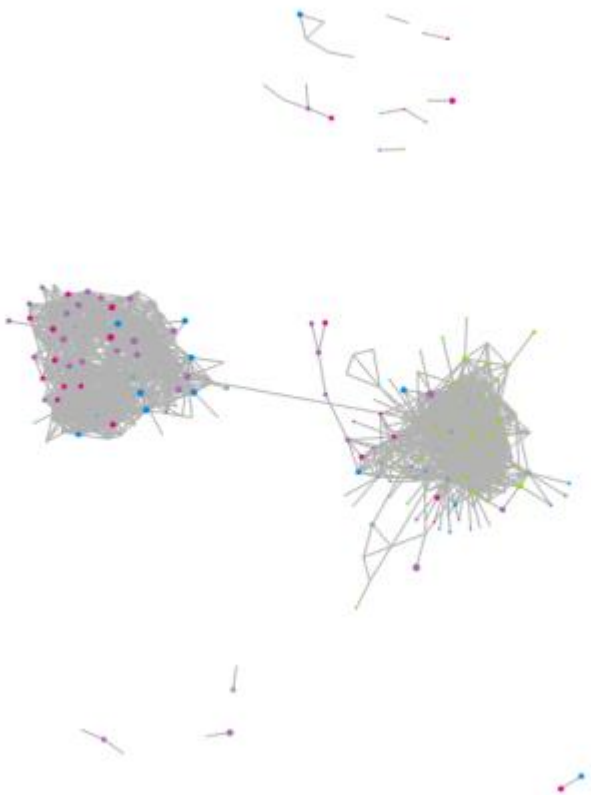


Рисунок 2.1 – Приклад соціального графу. Користувачі facebook

Соціальні мережі сприяють, по-перше, організації соціальних комунікацій між людьми, і по-друге – реалізації базових соціальних потреб. Можна виділити два трактування, які перетинаються між собою, це соціальна структурп та її спецефічна Інтернет реалізація.

Техніка соціометрії (описання соціальних груп в термінах соціальних графів) була вперше запропонована в роботах Дж. Морено. Термін “соціальна мережа” був введений в 1954 році соціологом Дж. Барнсом, але масового розповюдження термін набува починаючи з 2000 року, в зв’язку з надзвичайно швидким розвитком Інтернет технологій.

В наш час, багато вчених підтримують думку про існування гострого дефіциту систематичного представлення методів та алгоритмів мережевого аналізу, які були б придатні для сучасних прикладних досліджень.

Можна виділити наступні переваги від користування соціальними мережами:

– отримання інформації (в тому числі знаходження ресурсів) від інших членів соціальної мережі [19] (на рис. 2.2);

– верифікація ідей через участь у взаємодіях в соціальній мережі;

– соціальна користь від контактів (співучасть, самоідентифікація, соціальна рівність, соціальне сприйняття та інші);

– рекреація (відпочинок, та проведення часу). Вважається, що ключовими словами будь-якої соціальної мережі є: агент, думка агента, вплив та довіра, репутація.

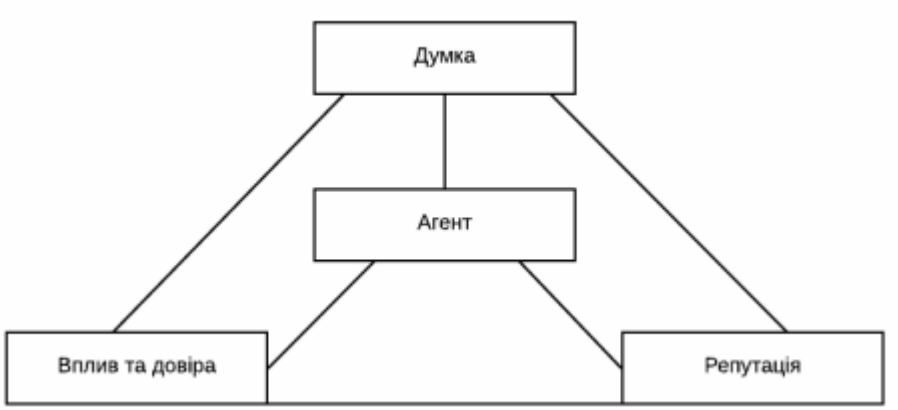


Рисунок 2.2 – Ключові слова соціальної мережі

По-перше, теорія графів надала словник понять для математичного опису властивостей соціальних структур. По-друге, теорія графів надає математичні інструменти для кількісного оцінювання характеристик соціальних мереж. По-третє, словник понять, математичні методи та теорія графів надають змогу доводити теореми про властивості соціальних структур [20].

В загальному випадку вершини графа відповідають акторам соціальної мережі, а ребра графа — комунікаціям між акторами. Розглянемо граф деякої соціальної мережі (на рис.2.3), який містить вершини $\boldsymbol{N = \{n_1, n_2, \dots, n_g\}}$ та ребра $\boldsymbol{E = \{1_1, 1_2, \dots, 1_L\}}$. Для комп'ютерного аналізу граф соціальної мережі представляють матриця суміжності, яку називають соціоматрицею. Як приклад показано соціоматрицю для графа:

\	$\boldsymbol{n_1}$	$\boldsymbol{n_2}$	$\boldsymbol{n_3}$	$\boldsymbol{n_4}$	$\boldsymbol{n_5}$	$\boldsymbol{n_6}$
$\boldsymbol{n_1}$	0	1	1	1	0	0
$\boldsymbol{n_2}$	1	0	1	0	0	0
$\boldsymbol{n_3}$	1	1	0	1	0	0
$\boldsymbol{n_4}$	1	0	1	0	0	0
$\boldsymbol{n_5}$	0	0	0	0	0	1
$\boldsymbol{n_6}$	0	0	0	0	1	0

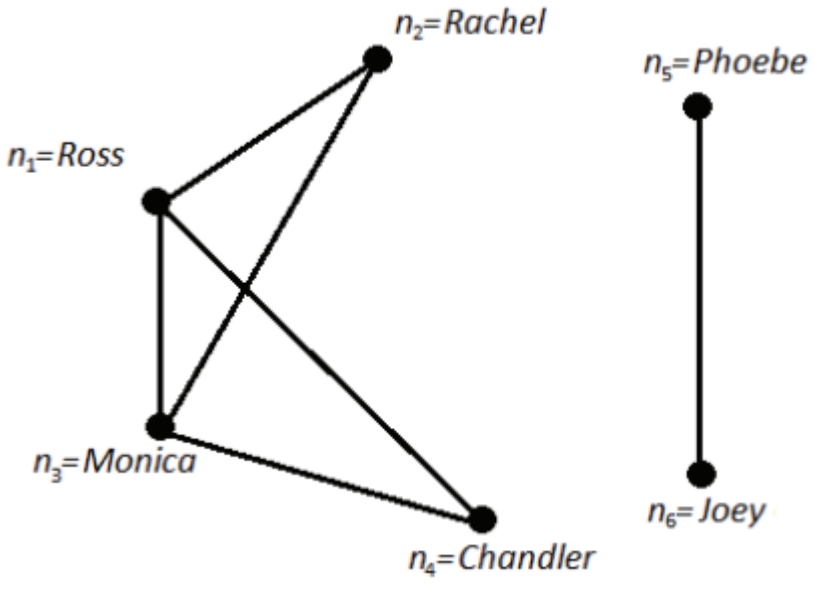


Рисунок 2.3 – Граф соціальної мережі

В аналізі соціальних мереж використовуються такі характеристики графів. *Ступінь вершини* ненаправленого графа, який дорівнює кількості ребер, що їй інциденті. Для направленого графа ступінь вершини розраховують окремо для вхідних та вихідних дуг. Ступінь вершини характеризує активність актора. Узагальненим показником для мережі є середній ступінь вершини:

$$\bar{\boldsymbol{d}} = \frac{\sum_{\boldsymbol{i=1}}^{\boldsymbol{g}} \boldsymbol{d}(\boldsymbol{n_i})}{\boldsymbol{g}} = \frac{\boldsymbol{2L}}{\boldsymbol{g}};$$

$$\bar{\boldsymbol{d}}_{\boldsymbol{1}} = \bar{\boldsymbol{d}}_{\boldsymbol{0}} = \frac{\boldsymbol{L}}{\boldsymbol{g}},$$

де $\boldsymbol{d}(\boldsymbol{n_i})$ - ступінь вершини $\boldsymbol{n_i}$;

$\boldsymbol{L}$ – кількість ребер/дуг в мережі;

$\boldsymbol{g}$ – кількість вершин в мережі;

$\bar{\boldsymbol{d}}_{\boldsymbol{1}}$ та $\bar{\boldsymbol{d}}_{\boldsymbol{0}}$ – середній ступінь вершини для вхідних та вихідних дуг відповідно.

Середній ступінь вершини використовується в багатьох стохастичних моделях як параметр активності мережі до створення зв'язків. Окрім середнього ступеню вершини використовується і оцінка його дисперсії:

$$\boldsymbol{S_D^2} = \frac{\sum_{\boldsymbol{i=1}}^{\boldsymbol{g}} (\boldsymbol{d}(\boldsymbol{n_i}) - \bar{\boldsymbol{d}})^{\boldsymbol{2}}}{\boldsymbol{g}},$$

де $\boldsymbol{d}(\boldsymbol{n_i})$ – ступінь вершини $\boldsymbol{n_i}$;

$\bar{\boldsymbol{d}}$ – середній ступінь вершин мережі;

g — кількість вершин у мережі.

Щільність графа — це характеристика зв'язаності графа, яка розраховується як відношення фактичної кількості ребер до їх максимально можливої кількості:

$$\Delta = \frac{\boldsymbol{L}}{\boldsymbol{g(g-1)/2}} = \frac{\boldsymbol{2L}}{\boldsymbol{g(g-1)}};$$

$$\Delta = \frac{\bar{\boldsymbol{d}}}{\boldsymbol{(g-1)}}.$$

Центральність — характеристика активності актора, яка розраховується як відношення ступеня відповідної вершини графа до кількості вершин, з якими можна утворити зв'язок:

$$\boldsymbol{C_D(n_i) = \frac{d(n_i)}{g-1}}.$$

Централізованість групи визначається так:

$$\boldsymbol{S_C^2 = \frac{\sum_{i=1}^{g}(C_D(n_i) - C_D)^2}{g}},$$

де $\boldsymbol{C_D(n_i)}$ – дисперсія центральності акторів.

Для регулярного графа централізованість буде рівна 0, та досягати максимуму при з'єднані типу «зірка».

Центральність актора може визначатись і у інший спосіб, наприклад, L. C. Freeman запропонував узагальнений індекс центральності та такі похідні показники як центральна близькість, центральність посередника та інформаційна центральність.

Престиж актора визначають відношенням кількості вхідних дуг відповідної вершини графа до кількості акторів в мережі:

$$\boldsymbol{P_D(n_i) = \frac{d_1(n_i)}{g-1}}.$$

Інколи престиж актора визначають за принципами PageRank-алгоритму з пошукової системи Google.

Однією з основних цілей застосування теорії графів є кластеризація соціальної мережі, тобто формалізоване виявлення в ній груп. Група — це фрагмент графа, щільність внутрішніх зв'язків в якому домінує над щільністю зовнішніх зв'язків. Малозв'язні групи, що мають розріджені зв'язки всередині, але водночас об'єднують декілька підгруп — є дуже критичними. Інформація в щільних групах, зазвичай, є схожою та надлишковою, тому для групи вона має меншу цінність, ніж та що проходить через слабкі зв'язки з інших підгруп [21].

L. C. Freeman припустив, що ідеальною підгрупою є повнозв'язна компонента графа (сильний альянс). Відповідно до таких припущень розроблено цілий ряд моделей підгруп:

n-clique — максимальний підграф, в якому відстань між двома будь-якими парами вершин не більше $\boldsymbol{n}$;

k-plex — максимальний підграф, який містить $\boldsymbol{g_s}$ вершин та має не менше як $(\boldsymbol{g_s - k})$ суміжних вершин у підграфі;

k-core — максимальний підграф, кожна вершина якого має не менше $\boldsymbol{k}$ суміжних вершин всередині підграфа;

LS-множина — підграф, в якому довільна множина вершин має більше внутрішніх ребер, ніж зовнішніх;

λ -множина — підграф, в якому міра зв'язності будь-якої пари вершин всередині підгрупи більша, ніж поза нею. Зв'язаність вершин $\lambda(\boldsymbol{i,j})$ визначається через мінімальну кількість зв'язків, які треба видалити, щоб вершини $(\boldsymbol{i,j})$ були недосяжні.

Модель спільнот. Ця модель базується на виявленні спільнот або груп користувачів з подібними інтересами, освітніми цілями або контекстами. Вона може допомогти учасникам мережі знайти людей зі схожими інтересами і обмінюватися досвідом [22].

Розглядаючи інструменти соціальних мереж для створення обговорень за визнач ною тематикою (Content), структуру віртуальної спільноти в соціальних мережах, доцільно розглядати як сукупність дискусій, створених за допомогою інструментів соціальних мереж, зареєстрованими користувачами (Members), які об'єднуються за ознакою мети, ідеологією спілкування та взаємодіють між собою не тільки в межах окремої дискусії, але з іншими дискусіями віртуальної спільноти та дискусіями інших віртуальних спільнот (Link). Таким чином, зовнішнє інформаційне середовище віртуальної спільноти – це сукупність віртуальних спільнот (сторінок

дискусій соціальної мережі, об’єднаних за ознаками інформаційного наповнення), агентів зовнішнього впливу (сторінки соціальної мережі, які не являються сторінками дискусій) та зв’язками між ними.

На рис. 2.4 відображено елементи зовнішнього інформаційного середовища в соціальних мережах , а саме:

Агенти зовнішнього впливу (Інтернет – ЗМІ, блоги політиків, відомих людей), які функціонують в соціальних мережах та є суб’єктами управління віртуальної спільноти щодо їх інформаційного наповнення та формування ідеології віртуальної спільноти. Агенти зовнішнього впливу будуть характеризуватися одностороннім зв’язком інформаційно-психологічного впливу на віртуальну спільноту.

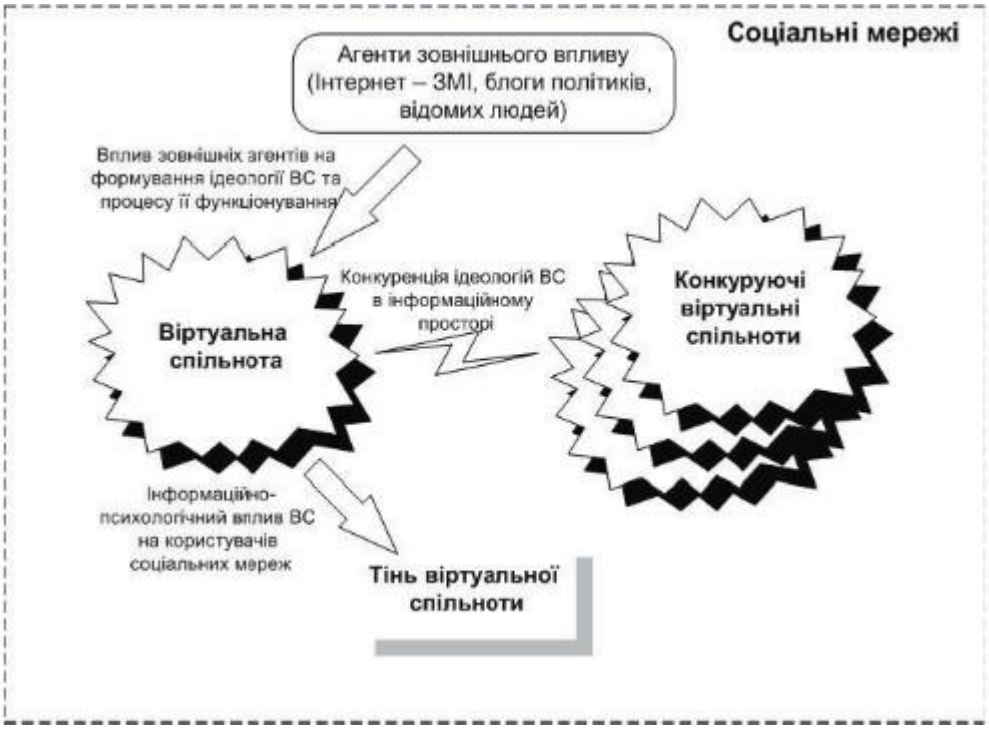


Рисунок 2.4 – Елементи зовнішнього інформаційного середовища в соціальних мережах

Віртуальні спільноти, які функціонують в інформаційному середовищі соціальних мереж з метою досягнення визначених цілей (деструктивного, конструктивного характеру) будуть характеризуватися наступними інформаційно-психологічними зв'язками [23]:

– одностороннім зв'язком з Агентами зовнішнього впливу, як об'єкт інформаційно-психологічного впливу;

– одностороннім зв'язком з тінню віртуальної спільноти, як суб'єкт інформаційно-психологічного впливу;

– двостороннім зв'язком з іншими віртуальними спільнотами, з метою конкуренції ідеологій віртуальних спільнот в інформаційному середовищі.

Тінь віртуальної спільноти – зареєстровані користувачі соціальних мереж, які не являються учасниками дискусій віртуальної спільноти та конкуруючих віртуальних спільнот, але зацікавлені ідеологією віртуальної спільноти.

Враховуючи визначення зовнішнього інформаційного середовища віртуальної спільноти та його елементів, модель зображено на рис. 2.5.

$$\begin{aligned}InfSpace = \langle & VirtualCommunity, \\ & AgentInfl, Shadow(VirtualCommunity), \\ & LinkExternal(VirtualCommunity), \\ & LinkExternal(AgentInfl)\rangle,\end{aligned}$$

Рисунок 2.5 – Склад віртуальної спільноти інформаційного середовища

VirtualCommunity – сукупність віртуальних спільнот в інформаційному середовищі.

AgentInfl – сукупність агентів зовнішнього впливу (Інтернет – ЗМІ, блоги політиків, відомих людей).

LinkExternal VirtualCommunity( ) – матриця зв’язків між віртуальними спільнотами в інформаційному середовищі.

LinkExternal AgentInfl( ) – матриця зв’язків між віртуальними спільнотами та агентами зовнішнього впливу в інформаційному середовищі.

Shadow(VirtualCommunity) – множина зареєстрованих користувачів соціальної мережі, які являються тінню віртуальної спільноти.

Внутрішнє інформаційне середовище віртуальної спільноти – це сукупність дискусій, які створюються зареєстрованими учасниками соціальної мережі і об’єднуються за ознакою мети, ідеологією існування та зв’язками між ними.

Рисунок 2.6 – Варіант структури внутрішнього інформаційного середовища віртуальної спільноти

Важливо враховувати, що написання освітньої соціальної мережі - це складний процес, який потребує великої кількості розробки, тестування та підтримки. Застосування сучасних технологій та моделей допоможе створити потужну та ефективну освітню платформу для спільного навчання та спілкування.

2.2 Технології розробки системи

Інтегроване середовище розробки Visual Studio Code

Visual Studio Code – це стартовий майданчик для написання, налагодження і складання коду, а також подальшої публікації додатків. Інтегроване середовище розробки (IDE) являє собою багатофункціональну програму, яку можна використовувати для різних аспектів розробки програмного забезпечення. Крім стандартного редактора і відладчика, які існують в більшості середовищ IDE, Visual Studio включає в себе компілятори, засоби автозавершення коду, графічні конструктори і багато інших функцій для спрощення процесу розробки. Інтерфейс зображено на рис. 2.7.

Крос-платформний. Це безкоштовний, з відкритим вихідним кодом і крос-платформний редактор, який працює на Windows, Linux і MacOS, так що ви можете працювати незалежно від платформи, на якій засновано ваш пристрій.

Підтримує безліч мов програмування. Коли ви відвідаєте веб-сайт Visual Studio Code, то миттєво зрозумієте, що він підтримує майже всі основні мови програмування. Він підтримує Python, JavaScript, HTML, CSS, TypeScript, C++, Java, PHP, Go, C#, PHP, SQL, Ruby, Objective-C та багато іншого.

Ви можете змінити мову для вибраного файлу. Він підтримує мову за промовчанням, яка буде відповідати вашому файлу, але ви також можете змінити мовний режим. Для цього натисніть на індикатор мови, розташований у правій частині рядка стану, який відкриє список “Вибір мовного режиму”. Тут можна вибрати різні мови для вашого поточного файлу.

Надає мовну документацію. Його веб-сайт містить документи, що стосуються спільних мов, які підтримує Visual Studio Code. Деякі з них - C ++, C #, CSS, Go, Python, PHP, Java та багато іншого.

Налагодження. VSCode поставляється з вбудованим налагоджувачем, який також є однією з його ключових функцій. Це допомагає прискорити цикл редагування, компіляції та налагодження будь-якого програміста. Проте за замовчуванням він поставляється тільки з відладчиком, який підтримує NodeJS, який може налагоджувати все, що переноситься на JavaScript, але знову ж таки, ви можете використовувати розширення для інших середовищ виконання.

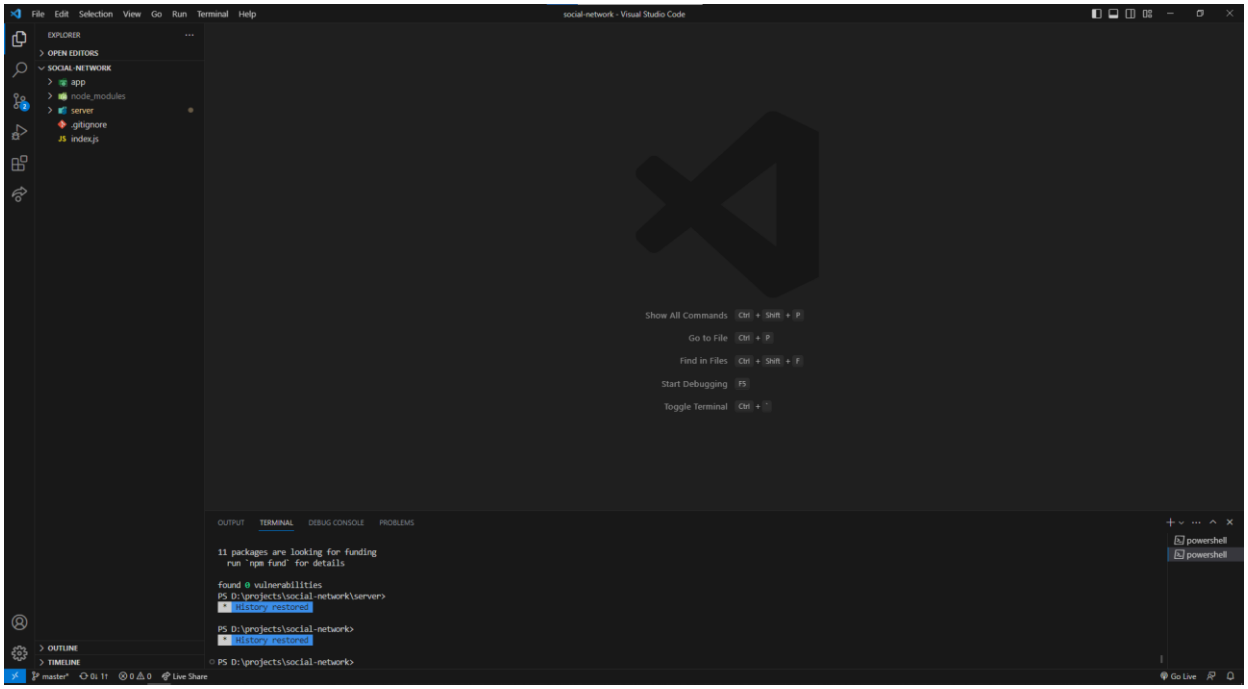


Рисунок 2.7 – Середовище розробки Visual Studio Code

Вбудована інтеграція Git. Visual Studio Code робить ще один крок уперед, надаючи повну інтеграцію Git, що дозволяє програмістам миттєво бачити зміни, не виходячи із редактора, можна переглянути на рис. 2.8. Ви можете знайти значок Git зліва від бічної панелі, де ви можете ініціалізувати його та виконати кілька команд Git, таких як pull, push, publish та інші. Крім того, VSC також працює з кількома репозиторіями Git, чи то локальними, чи віддаленими.

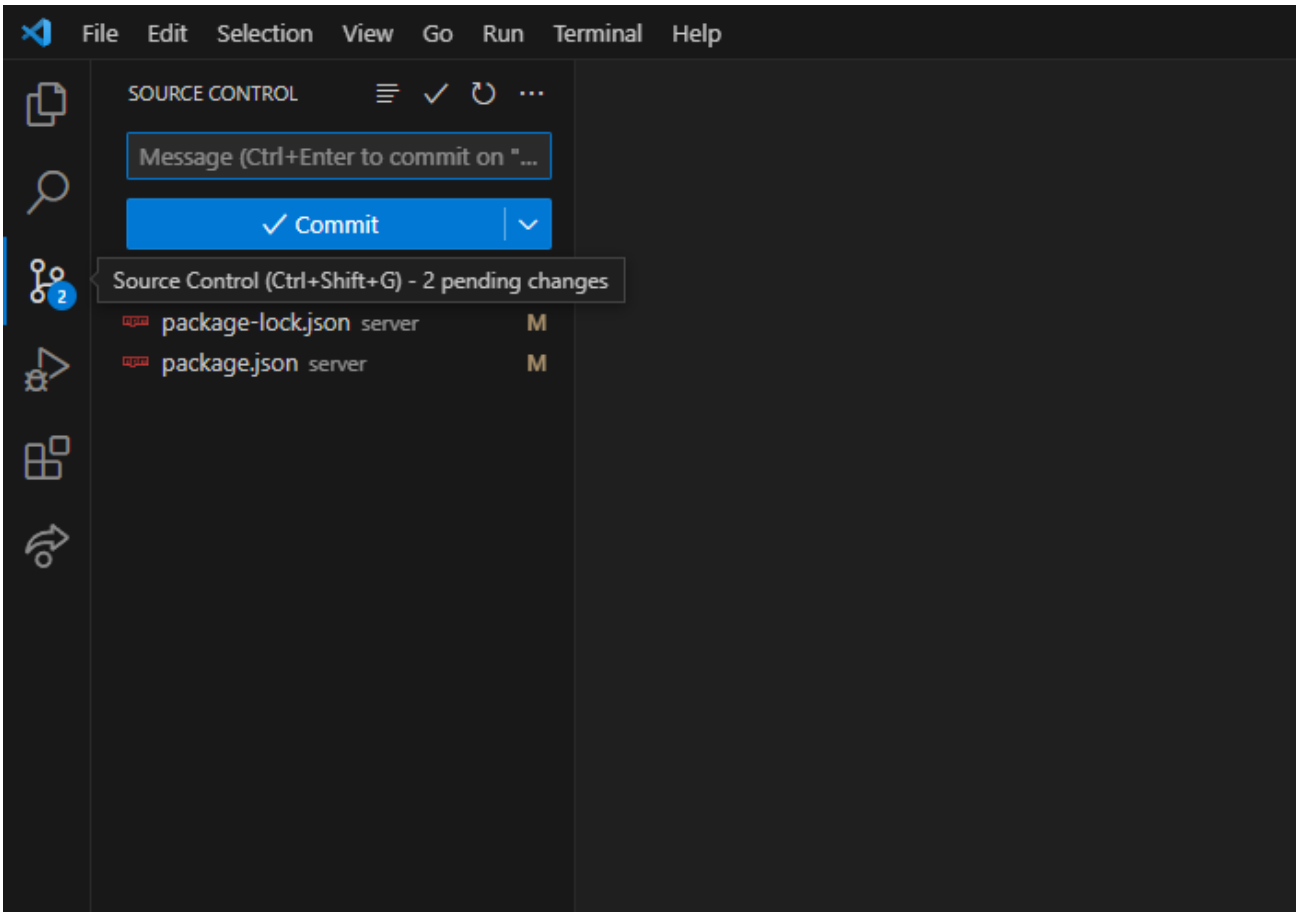


Рисунок 2.8 – Інтегрована Git частина

IntelliSense. Це функція, яка використовується програмістами для інтелектуального завершення коду, інформації про параметри, допомоги контенту, швидкої інформації та натяків на код. VSC надає IntelliSense для JavaScript, CSS,

HTML, TypeScript, JSON, Sass та інших мов програмування. Для інших мов ми можемо використовувати IntelliSense, додавши його розширення.

Палітра команд. Натискання Ctrl/Command+Shipt+P викликає панель команд, яка робить код VS доступним з клавіатури. Він дозволяє отримати доступ до всіх функцій VS Code, включаючи всі ярлики ключових слів. Крім того, ця палітра також дозволяє отримати доступ до багатьох команд.

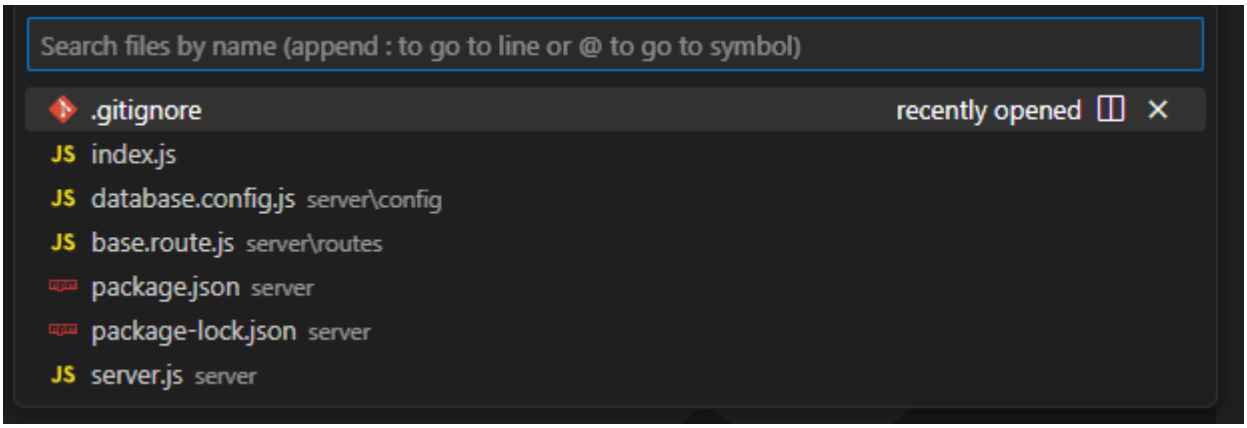


Рисунок 2.9 – Палітра команд в Visual Studio Code

Функції керування кодом. Visual Studio Code також надає функції управління кодом, такі як Go to Definition, Peek Definition, Find all References і rename Symbol. Клацнувши правою кнопкою миші у файлі коду, можна легко знайти ці функції в VSC.

Для виконання роботи було обрано основним фреймворком нашої мережі Angular.

Angular – це фреймворк для створення веб-додатків, розроблений компанією Google. Він забезпечує широкий спектр функцій для розробки додатків, включаючи структуру компонентів, директив, сервісів та модулів, а також механізми для маршрутизації, залежностей та обробки подій. Основною мовою програмування, яку використовують з Angular, є TypeScript, який є розширенням JavaScript.

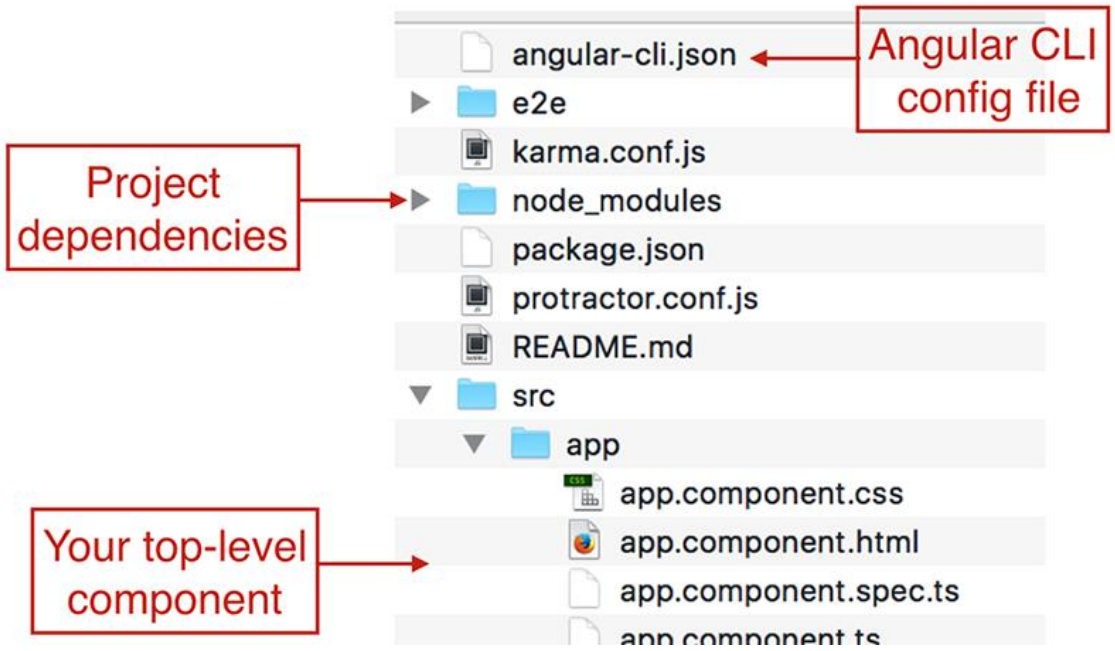


Рисунок 2.10 – Базова структура Angular додатку

Особливості Angular представлені нижче.

Компонентна архітектура: Angular заснований на компонентній архітектурі, що дозволяє розбити додаток на окремі компоненти, які можна повторно використовувати в різних місцях додатку [24].

Двостороння зв'язування даних: Angular забезпечує двостороннє зв'язування даних між представленням і моделлю даних, що дозволяє автоматично оновлювати дані в обох напрямках.

Директиви: Angular містить багато вбудованих директив, які дозволяють змінювати поведінку елементів DOM, таких як стилі, класи, атрибути і т.д.

Сервіси: Angular має механізм сервісів, що дозволяє розділяти код, що відповідає за бізнес-логіку та логіку доступу до даних, від коду, що відповідає за інтерфейс користувача.

Маршрутизація: Angular містить механізми для маршрутизації, що дозволяє керувати навігацією між сторінками додатка та передачі параметрів між ними.

Залежності: Angular має вбудований механізм впровадження залежностей, який дозволяє ін'єктувати сервіси та інші об'єкти в компоненти.

Тестування: Angular має вбудовані механізми для тестування додатків, що дозволяє перевіряти різні аспекти додатку, включаючи його функціональність, продуктивність та стабільність.

Анімації: Angular містить механізми для створення анімацій та переходів між сторінками, що дозволяє зробити додаток більш інтерактивним та привабливим для користувача.

Модулі: Angular дозволяє розбити додаток на модулі, що дозволяє розділити його на функціональні блоки та забезпечити кращу організацію коду.

Шаблони: Angular має вбудовані механізми для створення шаблонів, що дозволяє швидко створювати інтерфейс користувача та взаємодіяти з даними.

Підтримка веб-компонентів: Angular має підтримку веб-компонентів, що дозволяє використовувати компоненти, створені з інших фреймворків та бібліотек.

Швидкість та продуктивність: Angular дозволяє створювати додатки з високою продуктивністю та швидкістю завантаження завдяки використанню засобів оптимізації та компіляції коду.

Загалом, Angular є потужним та гнучким фреймворком для розробки веб-додатків, який забезпечує широкий спектр функцій та можливостей для розробки як простих, так і складних додатків.

TypeScript – це мова програмування, що розширює JavaScript та додає до нього деякі нові функції та можливості. Однією з головних особливостей TypeScript є те, що вона є статично типізованою мовою програмування. Це означає, що під час компіляції TypeScript перевіряє правильність типів даних, що використовуються в коді, та

попереджає про можливі помилки. Також, до його особливостей можна віднести наступні пункти [25]:

– підтримує класи, інтерфейси та інші об'єктно-орієнтовані конструкції, що дозволяють легко організовувати код та зменшувати його складність. Крім того, TypeScript має вбудовану підтримку асинхронного програмування, що дозволяє легко працювати з промісами, асинхронними функціями та іншими конструкціями;

– має можливість використовувати нові функції та можливості, які були додані в останніх версіях JavaScript, навіть якщо браузер не підтримує ці функції. Також TypeScript має вбудовану підтримку для модульної структури коду та можливість використовувати зовнішні бібліотеки;

– TypeScript є розширюваною мовою програмування, що означає, що розробники можуть створювати свої власні типи даних та інші розширення для мови. Це дає можливість розширювати функціональність TypeScript для виконання специфічних задач та дозволяє створювати більш складні програми з більш високим рівнем абстракції.

Бекенд буде написаний на **Node.js** – це відкрита платформа, що дозволяє виконувати JavaScript на сервері. Це означає, що Node.js може виконувати скрипти на стороні сервера, а не тільки на стороні клієнта в браузері. Node.js базується на двигуні V8, що розроблений Google для браузера Chrome, і використовує подійно-орієнтовану та неблокуючу модель введення-виведення, що дозволяє обробляти багато запитів одночасно.

До особливостей платформи можна віднести:

– вона дозволяє розробникам створювати високопродуктивні та масштабовані додатки. Завдяки своїй асинхронній моделі введення-виведення, Node.js може обробляти багато запитів одночасно, що дозволяє додатку працювати швидко та ефективно;

– велика кількість модулів та бібліотек, що доступні для використання. Завдяки цьому, розробники можуть швидко та легко створювати різноманітні додатки, використовуючи готові рішення та бібліотеки;

– Node.js також дозволяє використовувати JavaScript для розробки серверних додатків, що зменшує необхідність вивчати додаткові мови програмування та дозволяє розробникам працювати в єдиному екосистемі;

– Node.js має вбудовану підтримку WebSocket, що дозволяє створювати додатки реального часу, такі як онлайн-чати, онлайн-ігри та інші. Також Node.js має підтримку мультиплатформовості, що дозволяє розробляти додатки на різних операційних системах;

– є розширюваною платформою, що дозволяє розробникам додатків використовувати зовнішні модулі та бібліотеки, що дозволяє працювати з різноманітними інструментами та сервісами. Також Node.js має велику спільноту розробників, що розробляють та підтримують різні модулі та бібліотеки, що забезпечує швидкий розвиток та вдосконалення платформи;

– Node.js дозволяє розробникам створювати додатки за допомогою різних фреймворків, таких як Express.js, Koa.js, Meteor, Nest.js та інших, що дозволяють прискорити процес розробки та забезпечують швидку та легку розробку додатків;

– однією з особливостей Node.js є також можливість використання модуля Node Package Manager (NPM), який забезпечує доступ до тисяч бібліотек та модулів, що можуть використовуватись у проектах. Більшість цих бібліотек та модулів є безкоштовними та відкритими, що дозволяє розробникам ефективно використовувати готові рішення та зменшувати час на розробку.

В цілому, Node.js є потужною та ефективною платформою для розробки серверних додатків, яка дозволяє створювати високопродуктивні та масштабовані додатки, використовуючи JavaScript**.** Завдяки своїм особливостям та великій кількості модулів та бібліотек**,** Node.js дозволяє розробникам швидко та ефективно створювати

різноманітні додатки та взаємодіяти з іншими сервісами та інструментами.

У якості бази даних скористаємось **MySQL** – це безкоштовна система управління базами даних (СУБД) з відкритим вихідним кодом, яка дозволяє зберігати, організовувати та отримувати доступ до великих обсягів даних. MySQL є однією з найбільш популярних СУБД в світі та використовується для розробки веб-додатків та інших програмних застосунків [26].

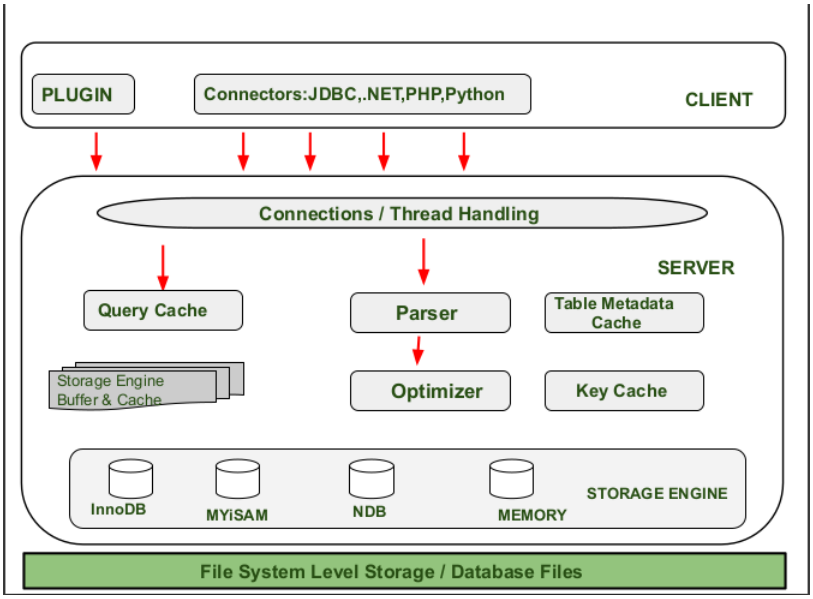


Рисунок 2.11 – Архітектура MySQL BD

Однією з основних особливостей MySQL є швидкість та продуктивність. Система дозволяє працювати з великою кількістю даних та виконувати запити до бази даних дуже швидко. MySQL також має досить простий та зрозумілий мову запитів SQL (Structured Query Language), яка дозволяє легко створювати та змінювати дані в базі даних.

MySQL також має велику кількість різноманітних функцій та можливостей, які дозволяють розробникам створювати складні та потужні застосунки. Серед цих можливостей можна виділити підтримку транзакцій, реплікацію даних, забезпечення

безпеки, індексацію даних та інші.

Ще однією з переваг MySQL є його гнучкість та можливість налаштування. Система може працювати на різних операційних системах та з різними мовами програмування, такими як PHP, Java, Python та інші. Крім того, MySQL може бути налаштований для роботи в різних конфігураціях та середовищах.

Він є потужною та ефективною СУБД, яка забезпечує швидкий та надійний доступ до великих обсягів даних. Завдяки своїм особливостям та можливостям, MySQL є популярним вибором для розробки веб-додатків та інших програмних засобів.

MySQL також дуже популярний у галузі веб-хостингу та хмарних сервісів. Багато провайдерів веб-хостингу пропонують підтримку MySQL як стандартну опцію, що дозволяє користувачам швидко створювати та керувати базами даних для своїх веб-сайтів.

Однією з найбільш популярних функцій MySQL є підтримка транзакцій. Це дозволяє розробникам зберігати цілісність даних та забезпечувати відновлення баз даних у разі непередбачуваних ситуацій. Транзакції дозволяють групувати запити до бази даних в один блок, який виконується як один цілісний процес. Якщо під час виконання транзакції сталася помилка, то всі зміни, зроблені в базі даних, відміняються.

Висновки до розділу 2

Написання освітньої соціальної мережі вимагає використання певних моделей, методів та інформаційних технологій. Освітні соціальні мережі відрізняються від звичайних соціальних мереж тим, що їх основна мета - забезпечення освітніх можливостей та співпраці між учнями, вчителями та іншими освітніми співробітниками. Деякі ключові аспекти, які потрібно враховувати при розробці

освітньої соціальної мережі, включають: модель користувача, методи навчання, спільноти і праці, оцінювання та відстежування прогресу, безпеку та конфіденційність.

Для реалізації освітньої соціальної мережі можна використовувати такі інформаційні технології:

Веб-розробка: Використання веб-технологій, таких як HTML, CSS, TypeScript та фреймворки для розробки фронтенду та бекенду додатку (Angular & NodeJS).

Бази даних: Використання баз даних для збереження користувачів, курсів, даних про прогрес тощо. Наприклад, можна використовувати реляційні бази даних, такі як MySQL.

Аутентифікація та авторизація: Застосування механізмів аутентифікації та авторизації для забезпечення безпеки та контролю доступу до ресурсів. В нашому випадку ми використаємо спосіб авторизація завдяки використанню JWT токену.

Інтеграція з інших систем: Можливість інтеграції з іншими освітніми системами або інструментами, наприклад, системами управління навчанням (LMS) або інструментами відеоконференцій.

Ці моделі, методи та інформаційні технології допомагають створити потужну та ефективну освітню соціальну мережу, яка сприяє навчанню, співпраці та розвитку освітньої спільноти.

3. МОДЕЛЮВАННЯ ТА ПРОЄКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

1.1 Опис вхідних даних та структури системи

Робота соціальної освітньої мережі являє собою постійне опрацювання вхідних та вже наявних даних. Уся її екосистема – це результат постійної роботи API (базою даних).

Вибір операційної системи припав на Windows. Вона буде керувати та надавати інформацію за усіма процесами, які виконуються в системі. Інформація, що надає нам операційна система не є відкритою. Але NodeJs має модуль os, який уможливлює нам перегляд хоста операційної системи, саму операційну систему, інформацію про кожне логічне ядро процесору, константи та багато багато іншого.

Основна частина виконання здійснюватиметься через текстові дані. Це найбільш важливий момент, тому що до нього можна віднести повідомлення, коментарі, імена, адреси електронної пошти, паролі. Тобто майже уся інформація, що опрацьовуватиметься – буде у вигляді тексту.

Числові дані потрібні будуть для встановлення дати реєстрації користувача, а також встановлення дедлайнів на виконання роботи. Ці процеси застосовують обчислення, тому робота з числовими даними буде необхідна.

Оскільки ми створюємо не лише соціальну мережу, а ще й освітню – ми маємо наділити наших користувачів можливістю відправки файлів: як для викладачів, так і для студентів. Тут нам знадобляться дані у вигляді файлів. В майбутньому ми будемо їх опрацьовувати, та оперувати шляхом взаємодії з базою даних.

Стуктуровані дані являють собою таблиці. В нашому випадку – це буде база даних MySQL. Це реляційна база даних, завдяки якій і виконуватиметься зміна та оновлення даних нашої мережі. Буде створено декілька таблиць, і поєднано їх

за різними типами зв’язку.

Нижче на рис. 3.1 переглянемо процес додатку до аналізу бази даних користувачів в загальному вигляді.

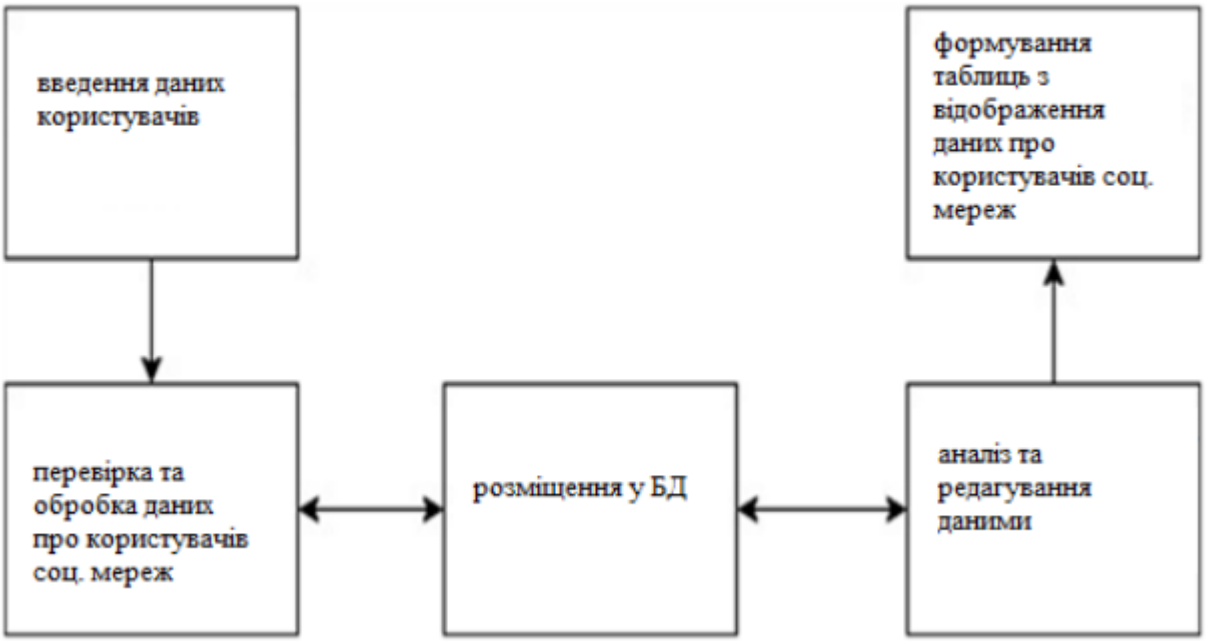


Рисунок 3.1 – Схема процесу додатку до аналізатора бази користувачів у загальному вигляді

Діаграма класів служить для представлення статичної структури моделі системи в термінології класів об'єктно-орієнтованого програмування.

Діаграма класів може відбивати, зокрема, різні взаємозв'язки між окремими сутностями предметної області, такими як об'єкти і підсистеми, а також описує їх внутрішню структуру (поля, методи тощо) і типи відносин (спадкування, реалізація інтерфейсів тощо).

На рис. 3.2 не вказується інформація про тимчасові аспектах функціонування системи. З цієї точки зору діаграма класів є подальшим розвитком концептуальної моделі проектованої системи. На цьому етапі принципово знання

ООП підходу і патернів проектування.

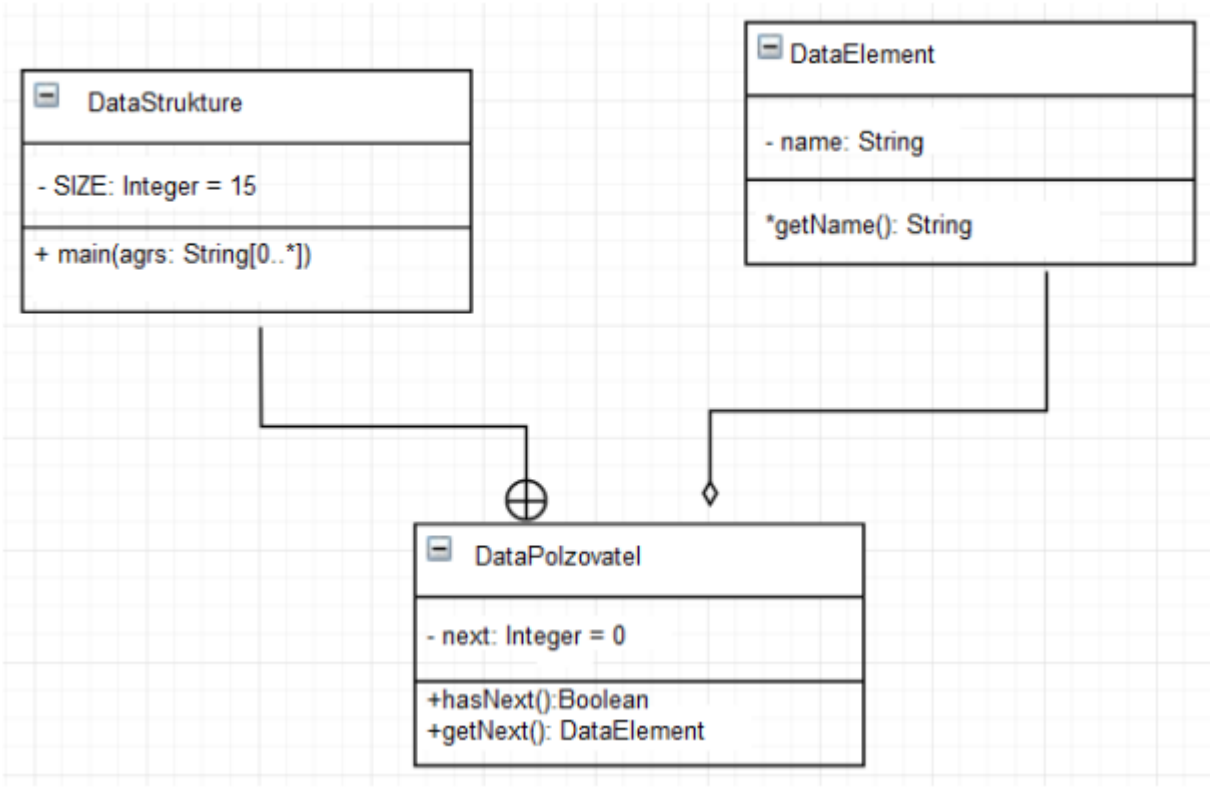


Рисунок 3.2 – Діаграма класів використання додатку бази користувачів

Діаграма компонентів, описує особливості фізичного представлення системи. Діаграма компонентів дозволяє визначити архітектуру розроблюваної системи, встановивши залежності між програмними компонентами, в ролі яких може виступати вихідний, бінарний і виконуваний код.

У багатьох середовищах розробки модуль або компонент відповідає файлу. Стрілки, що з'єднують модулі, показують відношення взаємозалежності, аналогічні тим, які мають місце при компіляції початкового програмного коду. Основними графічними елементами діаграми компонентів є компоненти, інтерфейси і залежності між ними.

1.2 Проектування інформаційної системи

Проектування інформаційної системи (ІС) є важливим етапом у розробці будь-якої інформаційної системи. Воно спрямоване на визначення архітектури, функціональності та характеристик системи з метою задоволення потреб користувачів та досягнення поставлених цілей організації. Проектування ІС визначає, як система буде працювати, які функції будуть доступні, які дані будуть оброблятися та як будуть взаємодіяти її компоненти.

Проектування ІС дозволяє детально вивчити та зрозуміти потреби та вимоги користувачів, бізнес-процесів та організації загалом. Це допомагає побудувати систему, яка відповідає цим вимогам та забезпечує їх задоволення.

Дозволяє створити систему, яка працює ефективно та ефективно, забезпечуючи оптимальне використання ресурсів та досягнення мети організації. Також визначає, як компоненти системи будуть взаємодіяти між собою, сприяючи покращенню спілкування, обміну даними та спільній роботі в рамках організації.

Узагальнену архітектуру додатку, який включає в себе засоби збереження даних та маніпулювання даними (сервер БД), сервер, робочу станцію, а також сукупність програмних модулів, що реалізують обробку даних, представлено на рисунку нижче.

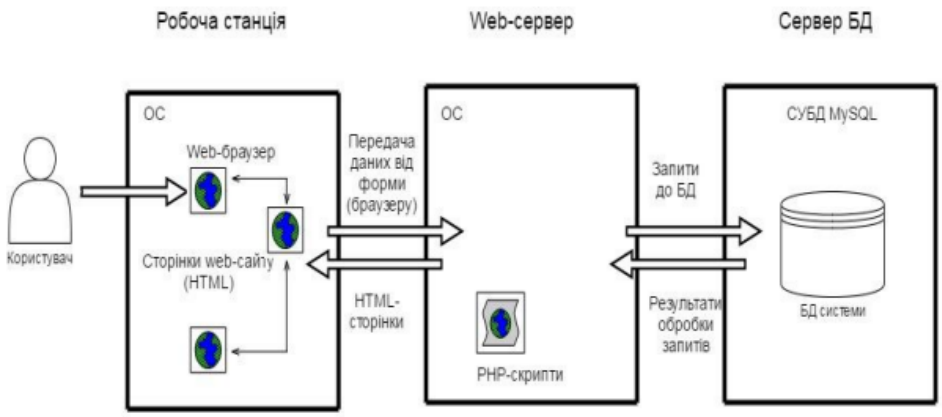


Рисунок 3.3 – Загальна архітектура інформаційної системи

Додаток для користувачів соціальної мережі повинен поєднувати систему із застосуванням елементів та даних проекту, що зберігаються у БД, як на рис.3.4. Для зручності користувача повинен бути створений ергономічний інтерфейс.

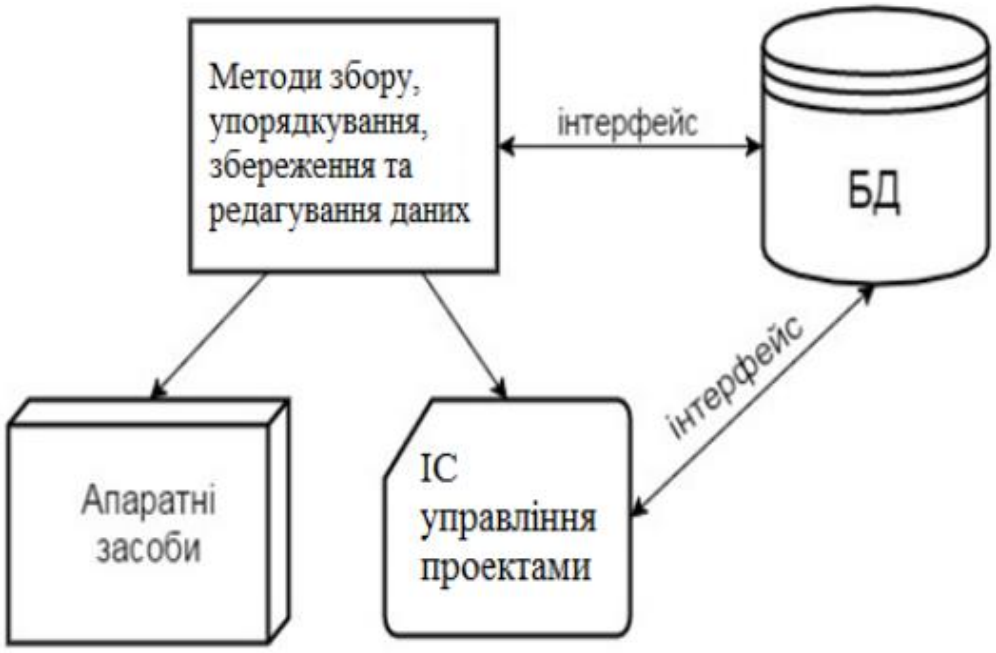


Рисунок 3.4 – Структура додатку для користувачів соціальної мережі

Основними вимогами до додатку та його системи є повнота, охоплення всіх сторін інформаційного, програмного, технічного забезпечення, які зустрічаються в процесі експлуатації системи.

Система повинна бути комплексною. Основні переваги такого додатку у порівнянні з традиційними методиками, полягають у можливостях спільного аналізу великих груп даних і їх взаємозв’язку. Проектований додаток повинен імітувати технологію досліджень. Система додатку повинна бути відкритою, забезпечуючи легкість модифікації і експлуатації до нових умов для підтримки її на сучасному рівні.

Для збереження, візуалізації та аналізу користувачів соціальної мережі було розроблено додаток для обробки даних, що зберігається у БД. Для зручності користувача розроблюється ергономічний інтерфейс. Просторові дані та їх зв'язки із табличними даними погано описуються реляційної моделлю, тому повна модель даних в додатку (рисунок 3.10) має змішаний характер, тоді як семантична інформація об'єктів буде представлена реляційними таблицями.

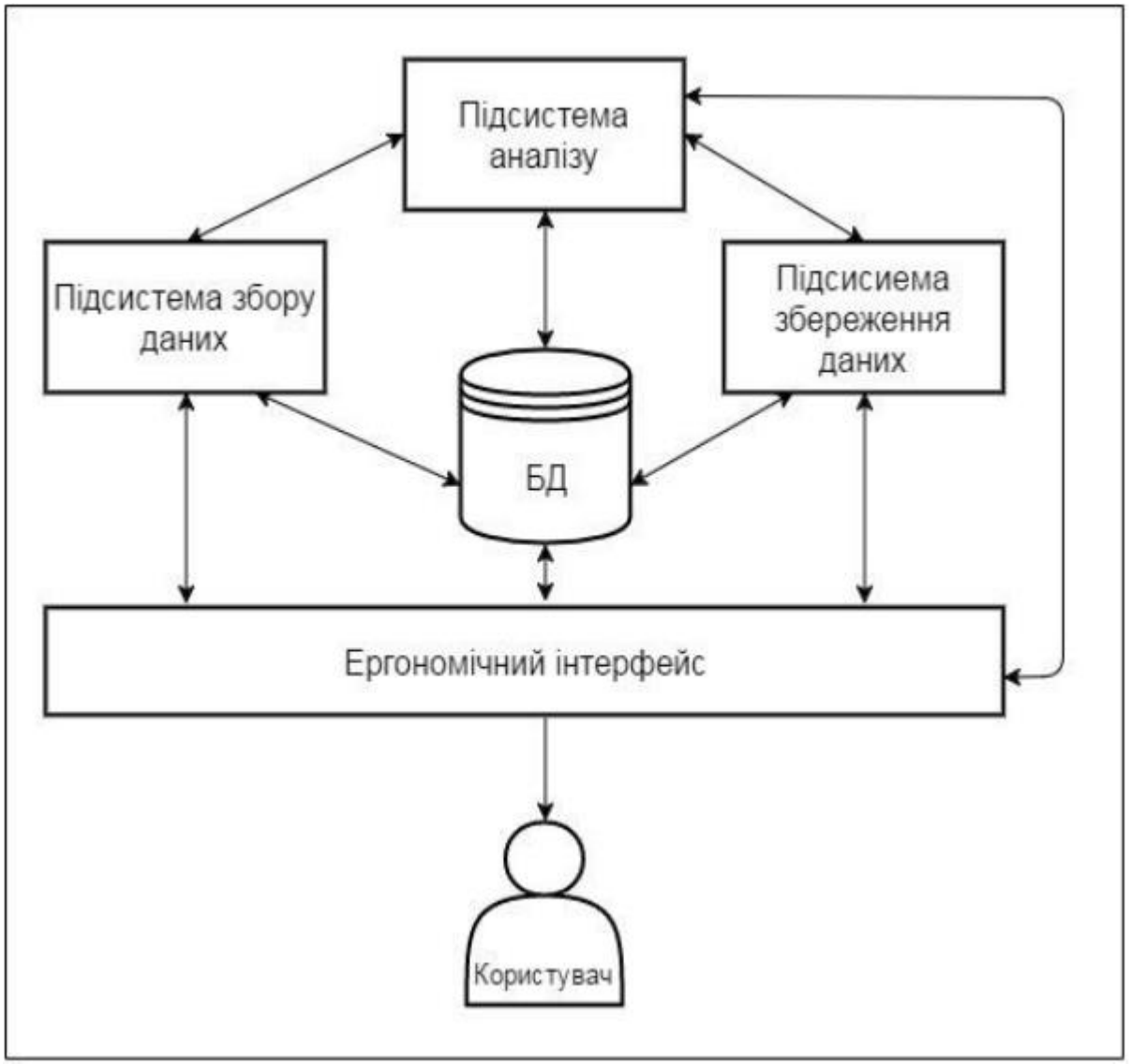


Рисунок 3.5 – Структура додатку бази користувачів соціальної мережі

Використовувана модель даних аналізатора повинна задовольняти такі умови:

– БД повинна легко розширюватися при реорганізації та розширенні предметної зони;

– БД має легко змінюватися при зміні програмного та апаратного середовища;

– дані до включення у БД слід перевіряти на достовірність;

– доступ до даних, які розміщуються в БД, повинні мати лише особи з відповідними повноваженнями;

– дані мають розміщуватись у форматах, доступних для розробленого додатку-аналізатора.

Основний функціонал мережі буде розподілено усього на 3 ролі: адміністратор, викладач та студент. Кожна з них матиме свій власний спектр прав. Інтерфейс також буде створено відповідно. Розглянемо їх кожного окремо та більш детально.

Administator – користувач, що матиме усі можливі права у нашій мережі. Перший адміністратор призначається розробником і далі матиме можливість додавати ролі адміністратора та викладача будь-якому юзеру. Також він може створювати нові групи, додавати до них студентів а також прив’язувати ці групи до дисципліни викладача. Адміністратор може змінювати контент на вкладці з домашнім завданням, завантажувати власний файл.

Teacher – користувач, що матиме схожі права, що і адміністратор, за винятком можливості зміни ролі користувача та його видалення.

Student – користувач, який спершу не матиме можливості приєднуватись до наявних груп. Лише адміністратор та викладач зможуть затвердити його належність. У студента є можливість змінювати деякі дані свого профілю а також переглядати та завантажувати домашні завдання.

Визначення та запис ролі користувача буде відбуватися у процесі аутентифікації. Для початку користувач реєструється, для цього він матиме заповнити наступні поля:

– логін – це поле має буде сформовано наступним чином: перший рік навчання_груп. Наприклад: 19_101. Валідація цього поля відбуватиметься завдяки регулярному виразу, що буде сформовано додатково;

– електронна адреса – поле, що буде валідуватися завдяки Angular. На пошту буде висилатись посилання, по проходженню на яке підтвердиться пошта.;

– пароль – це надійсніть та збереженність даних користувача. Він захищатиме його від зловмисників.

JWT token

У нашому додатку, для постійної перевірки а також затвердження особи будемо використовувати технології JSONWebToken. Це один з найкращих варіантів аутентифікації даних, які можна зберігати у самому токені.

Структура токену складається з трьох частин (рис. 3.6):

– заголовку (header) - рядок, що закодовує звичайний JavaScript об'єкт, який описує токен, а також використаний алгоритм хешування;

– навантаження (payload) - формує основу токена. Довжина корисного навантаження пропорційна кількості даних, збережених у JWT. Загальне правило: зберігати мінімум у JWT;

– та підпису (signature). згенерованого на основі заголовка та корисного навантаження. Використовується для перевірки JWT.

У логіку нашої мережі закладено мануальне призначення ролей для викладачів. За замовчуванням, до нашої БД ми додаємо кілька користувачів, які матимуть роль Administrator. Усі наступні зареєстровані користувачі за замовчуванням будуть отримувати роль student, адміністратори можуть скористатися

пошуком користувача, щоб знайти профіль і вже згодом змінити у ньому роль на

teacher для викладачів.

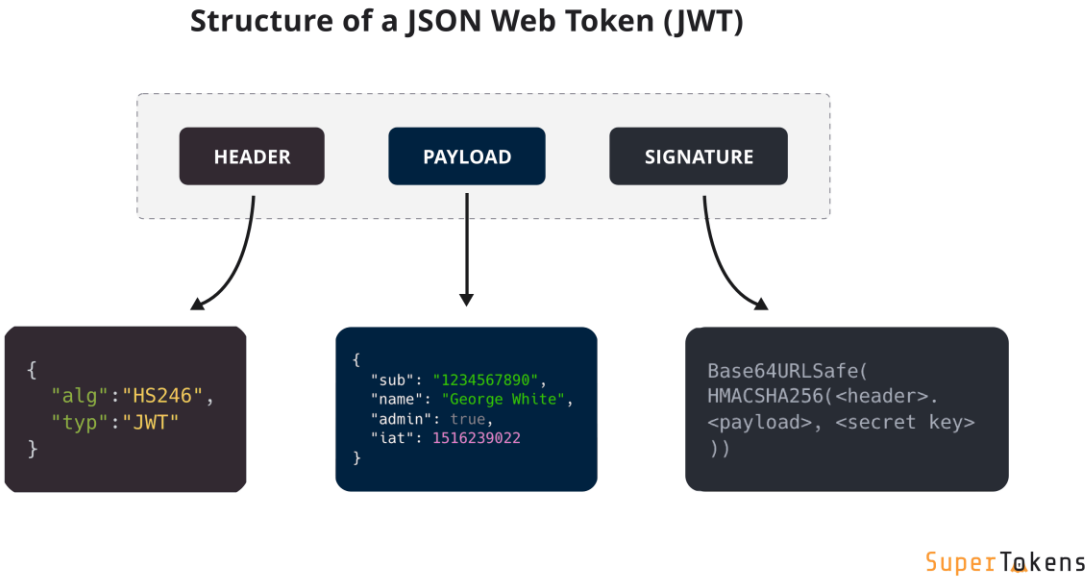


Рисунок 3.6 – Структура JWT

Власне процес реєстрації та подальшої авторизації не є складними у їхній реалізації. Найважливішим питанням є процес аутентифікації користувача. Для цього нам необхідно буде задіяти БД для оперування з даними. На цьому етапі ми маємо сформулювати наші таблиці, їх поля та типи цих полів.

Тому, користуючись базою даних MySQL, усі дані нашої мережі мають в автоматичному режимі записуватися до бази даних під час реєстрації, а також зчитуватись під час перевірки наявності користувача з ФІО та поштою або перевіряти дані для валідації під час входу до аккаунту.

Guards

Задля обмеження переміщення користувача по мережі у Angular було створено Guards. Завдяки функціоналу цього інструменту ми можемо в залежності від

виконуваності умови дозволяти чи забороняти користувачу переміщуватись на інші роути.

Наприклад в нашій мережі буде створено AuthGuard. Даний guard перевірятиме наявність JWT токену, і в залежності від його даних буде дозволяти користувачеві переміщуватись по різним вкладкам мережі. Загальний приклад роботи guard зображено на рис. 3.7.

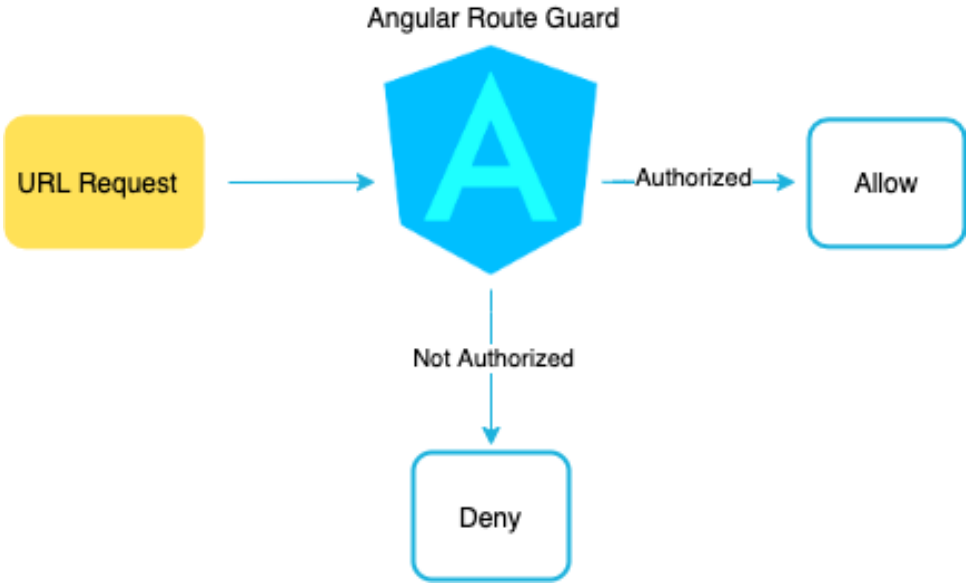


Рисунок 3.7 – Схема роботи guard в Angular

Логіка процесу входу до аккаунту описана нижче:

–guard адресує користувача на сторінку логіна;

–у разі, якщо користувач ще не зареєстрований, він може перейти на форму реєстрації;

–після заповнення усіх полів форми а також за умови унікальності електронної пошти та ФІО користувача, його дані буде додано до БД;

–нарешті користувач зможе ввести свій логін та пароль, після чого відбудеться аутентифікація та валідація його даних;

–у разі успішної перевірки буде створено закодований JWT токен, guard зможе «допустити» користувача до основного контенту освітньої соціальної мережі.

Детальну схему процесу авторизації наведено нижче:

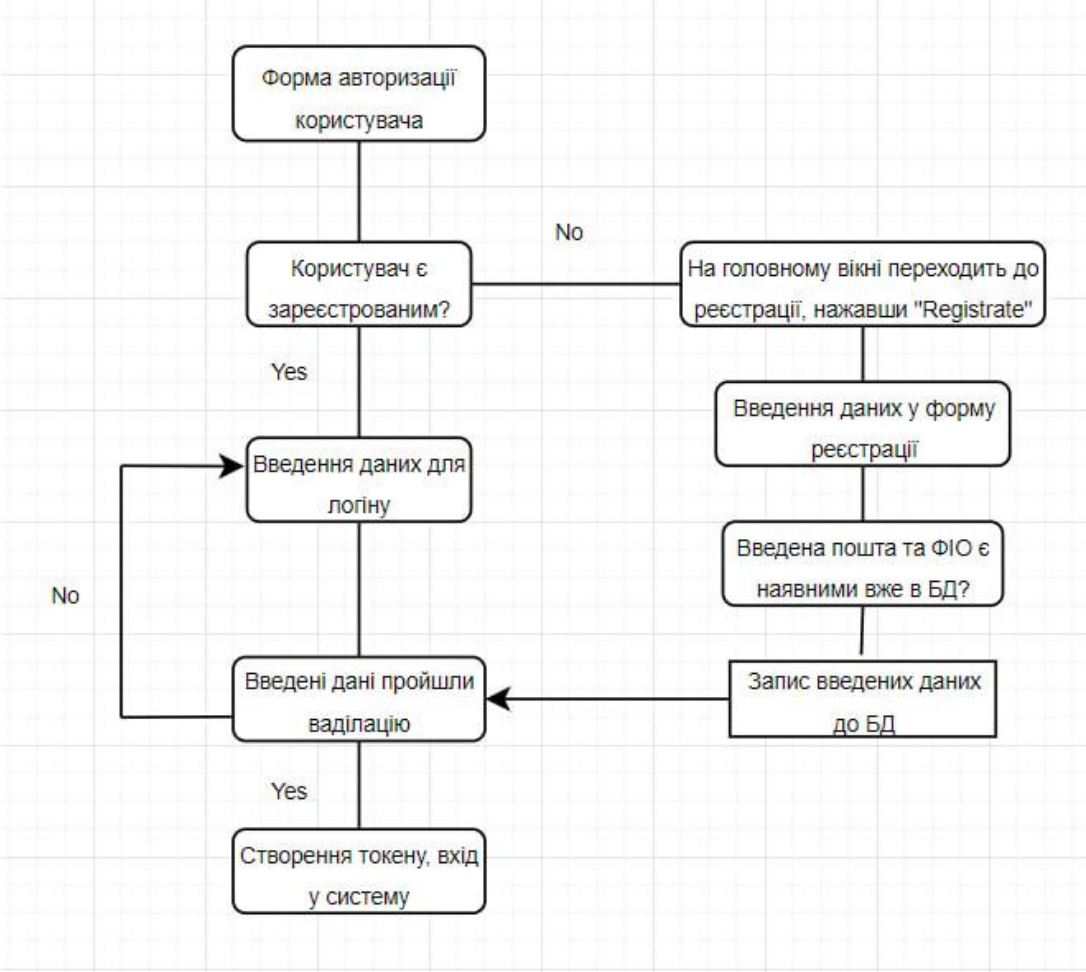


Рисунок 3.8 – Схема логіки системи авторизації

Модульна архітектура.

В світі web-розробки модульна архітектура Angular – це дещо особливе. Можливо, це одна з таких ідей, яка дається не усім новачкам але вона того варта.

Сама складність полягає в тому, що у web-розробці вже використовується модульна архітектура. І тут мова йде про ES6-імпорти.

Так як Angular-модулі додають в систему додатковий рівень логістичного групування, важливо, щоб їхня структура як можна краще відповідала задачам, що вирішуються з їх допомогою [27].

Одна з головних цілей проектування архітектури додатку – спростити розуміння та підтримку коду за рахунок існування чіткої цілі у кожного окремого модуля та логічних взаємозв'язків між модулями.

Головний кореневий модуль – RootModule, використовується у якості точки входу. Він декорований за допомогою @NgModule. Поглянемо на його стандартний вигляд:

```
// app.module.ts
@NgModule({
  imports: [BrowserModule],
  declarations: [AppComponent],
  bootstrap: [AppComponent],
})
export class AppModule { }
```

В якості аргументу в декораторі @NgModule використовується JavaScript об'єкт.

У властивості declarations ми оголошуємо компоненти, які містить наш модуль. В даному випадку це компонент AppComponent, Компонентів може бути декілька, вони оголошуються через кому (як в звичайному JavaScript масиві).

```
declarations: [AppComponent]
```

Модульна архітектура соціальної мережі буде логічно поділена від кореневого модуля на дві частини: модуль авторизації а також модуль з контентом (рис. 3.9). І далі, по ієрархії, вони ділитимуться на менші модулі, які включатимуть в собі наступні елементи проекту. Разом вони немов цеглини складаються воєдино і ми отримуємо діючий додаток. На рисунку нижче зображено схему розподілу модульної архітектури

в нашій мережі.

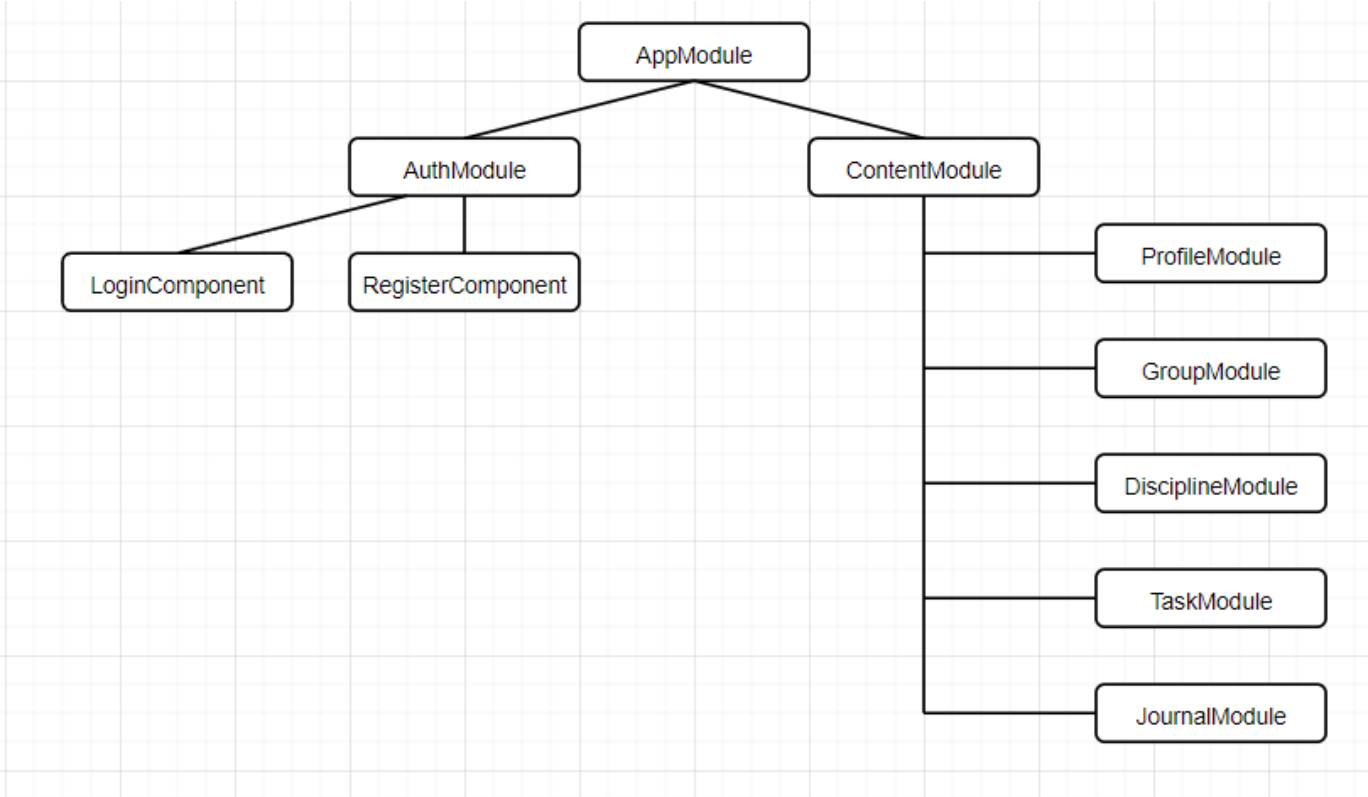


Рисунок 3.9 – Модульна архітектура мережі

Databases.

Архітектура MySQL дуже відрізняється від архітектур інших серверів баз даних, що робить цю СУБД корисною для одних цілей, але одночасно не дуже підходящим вибором для інших. MySQL не є ідеальною, але вона досить гнучка для того, щоб добре працювати в дуже потребуючих середовищах, наприклад у web-додатках [26].

В той же час MySQL дозволяє застосовувати вбудовані додатки, сховища даних, індексування вмісту, програмне забезпечення для доставки, високонадійні системи з резервуванням, опрацювання транзакцій в реальному часі (OLTP) та багато іншого.

Для того, щоб ефективно використовувати MySQL, потрібно розібратись в її устрої. Гнучкість системи проявляється у багатьох аспектах. Наприклад, ми можемо

налаштувати її для роботи на різноманітному обладнанні і підтримці різних типів даних. Але самою незвичайною і важливою особливістю MySQL є така архітектура підсистеми збереження, в якій опрацювання запитів і інші серверні задачі відокремлені від збереження та вилучення даних. Подібний розподіл задач дозволяє обирати спосіб збереження даних, а також налаштовувати продуктивність, ключові характеристики і т.п.

Аби добре розуміти роботу сервера, треба мати уявлення про взаємодію його компонентів. На рисунку нижче показано логічний вигляд архітектури MySQL.

Рисунок 3.10 – Логічний вигляд архітектури серверу MySQL

На верхньому рівні розташовані служби, які не є унікальними компонентами MySQL. Вони необхідні більшості мережевих клієнт-серверних інструментів або серверів: для обслуговування з’єднань, аутентифікації, забезпечення безпеки і т.п.

Другий рівень набагато цікавіший. Тут знаходиться більша частина «мізків» MySQL: код для опрацювання, аналізу, оптимізації та кешування запитів, а також усі

вбудовані функції. Також тут знаходяться усі інструменти, що використовуються у підсистемах збереження, наприклад процедури, тригери та представлення.

Третій рівень вміщує підсистеми збереження даних. Вони відповідають за збереження усіх даних в MySQL та їх отримання. Подібно різноманітним файловим системам, доступним для GNU/Linux, кожна підсистема збереження даних має як сильні так і слабкі сторони. Сервер взаємодіє з ними через API підсистеми збереження даних. Цей інтерфейс приховує приховує відмінності між такими підсистемами і робить їх практично прозорими на рівні запитів. API містить пару десятків низькорівневих функцій, які виконують операції типу «почати транзакцію» або «отримати рядок з таким первинним ключем». Підсистеми збереження не аналізують запити MySQL і не взаємодіють одне з одним, вони лише відповідають на вихідні від сервера запити.

Важливим аспектом при роботі з базою даних є розгляд та розуміння зв’язків між таблицями. Зв’язки відбуваються за допомогою ключів. Наприклад, у створеній нами раніше таблиці користувачів є первинний ключ – поле id. Якщо ми захочемо зробити таблицю зі статтями і зберігати в ній авторів цих статей, то ми можемо додати нову колонку author_id і зберігати в ньому id користувачів із таблиці users.

Це був лише один з прикладів. Усього ж типів подібних зв’язків може бути три:

– один-до-одного;
– один-до-багатьох;
– багато-до-багатьох;

При зв’язку один-до-одного кожному запису таблиці відповідає лише один запис у іншій таблиці.

В нашому випадку буде створено таблицю, в якій зберігатиметься профіль користувача. У ньому можна вказати інформацію про себе, особисті данні, емейл,

змінювати вже створений пароль.

Варто приділити увагу такому оператору SQL - INNER JOIN. Він використовується для об’єднання рядків з двох та більше таблиць. Для запиту використовується наступний синтаксис:

```
SELECT колонки FROM таблиця1 INNER JOIN таблиця2 ON умова для зв’язку
```

Зв’язок один до багатьох є самим поширеним типом зв’язку між таблицями бази даних. Уявимо, нам потрібно реалізувати певну БД, яка веде облік даних про користувачів. У користувача є: ім’я, фамілія, вік, номера телефонів. При цьому у кожного користувача може бути від одного і більше номерів телефону.

В такому випадку ми можемо спостерігати наступне: користувач може мати багато номерів телефонів, але не можна сказати, що номеру телефону належить певний користувач.

Іншими словами, телефон належить лише одному користувачу. А користувачу можуть належати 1 і більше телефонів.

Приблизно так і можна пояснити сутність зв’язку один-до-багатьох.

Зв’язок багато-до-багатьох виникає, коли більшість рядків однієї таблиці відповідають більшості рядків іншої таблиці. Щоб зв’язати ці дві таблиці між собою, потрібно створити третю таблицю, створивши з кожною з перших двох зв’язок один-до-багатьох.

Тепер більш детально розглянемо створені таблиці до нашої бази даних а також зв’язки між ними.

Першою буде таблиця, що міститиме у собі дані про користувача після проходження реєстрації. Автоінкрементом користувачеві буде призначено унікальний id, буде додано вказані ім’я, фамілію, пошту, пароль та номер групи. Як було зазначено вище, роль користувачеві за замовчуванням буде призначено як student, і лише адміністратор зможе змінити її на teacher.

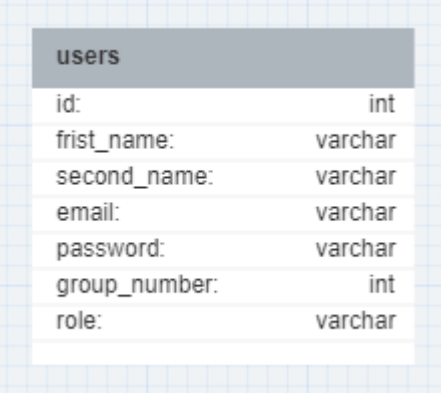


Рисунок 3.11 – Склад таблиці з користувачами

Друга таблиця у нас вміщуватиме у собі інформацію про існуючі дисципліни. Вона матиме наступні поля: унікальний номер кожної дисципліни, айді викладача, який її створив, список груп у json-форматі, які матимуть цю дисципліну як спільну, а також назву та опис дисципліни.

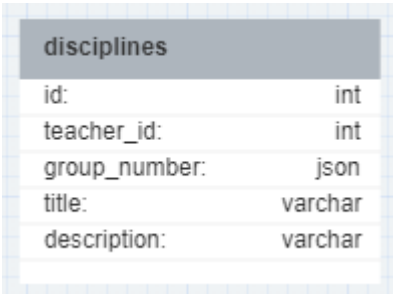


Рисунок 3.12 – Склад таблиці дисциплін

Третя таблиця буде про окремі завдання (лабораторні роботи) для кожної дисципліни. Кожне завдання матиме унікальний номер, айді дисципліни, аби їх можна було пов’язати і отримати дані викладача, що веде цей предмет, назву та опис конкретного завдання а також місце для завантаження файлу з завданням (тип blob).

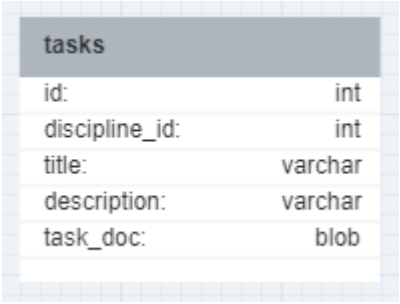


Рисунок 3.13 – Склад таблиці з завданнями для студентів

У четвертій таблиці будуть поміщені усі завантаженні завдання студентів. Таблиця буде складатись з: унікального номера домашнього завдання, айді студента, що виконав, айді завдання, на яке завантажено виконання, статус виконання завдання, оцінку а також студент зможе помічати завдання як виконане, для спрощення візуального сприйняття.

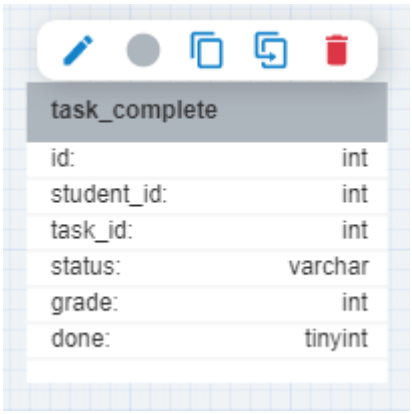


Рисунок 3.14 – Склад таблиці з виконаними завданнями студентів

У п’ятій таблиці зазначено журнал оцінок студента. Ми матимемо номер для кожної оцінки, айді студента, якому ця оцінка належить а також айді зданої роботи, щоб потім через неї можна було отримати та підрахувати усі бали по кожній з дисциплін.

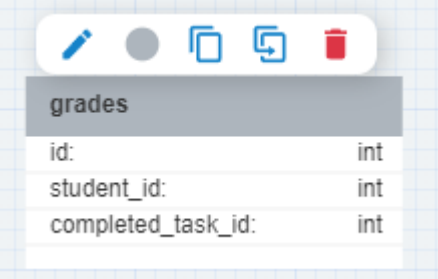


Рисунок 3.15 – Склад таблиці з оцінками

Останньою буде таблиця, яка відповідатиме за відображення листування авторизованого користувача з іншими. У цій таблиці буде унікальний номер повідомлення, айді від кого буде повідомлення, айді кому буде повідомлення і власне саме повідомлення типу json, аби користувачі могли надсилати не лише текст, а і файли різного типу одне-одному.

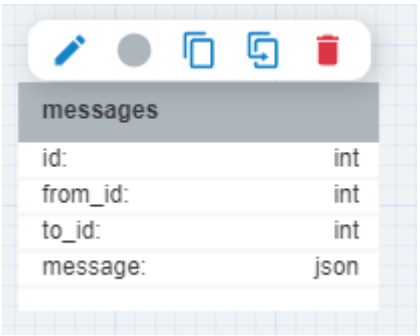


Рисунок 3.16 – Склад таблиці з повідомленнями

Перший зв’язок між таблицями буде між дисциплінами та користувачами, щоб зв’язати викладача зі списку користувачів та дисципліною, яку він створюватиме.

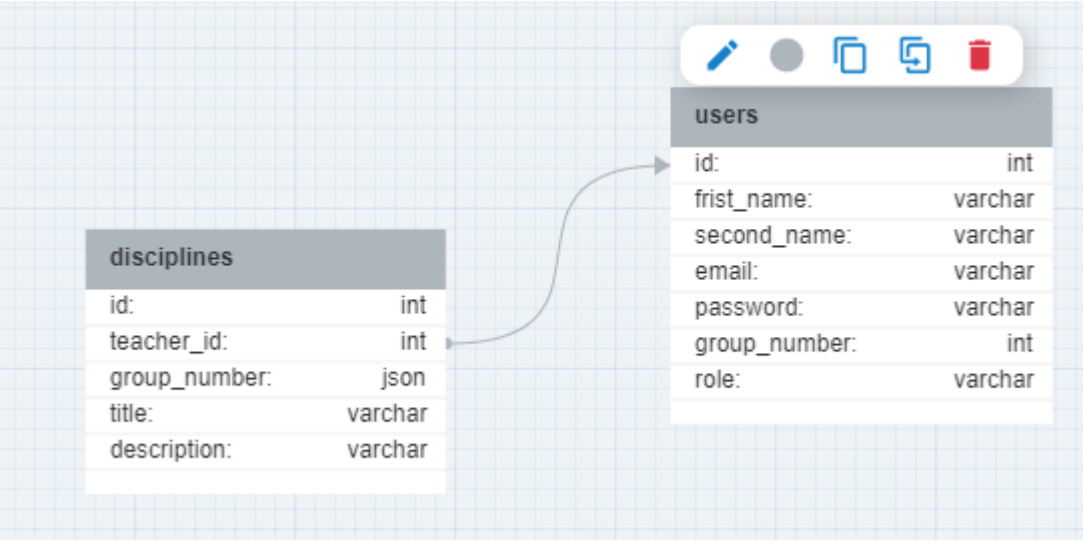


Рисунок 3.17 – Зв’язок один-до-одного дисциплін та користувачів

Другий зв’язок показано між завданнями та дисциплінами. Де декілька створених розділів з завданням можуть відповідати лише одній дисципліні за типом один-до-багатьох.

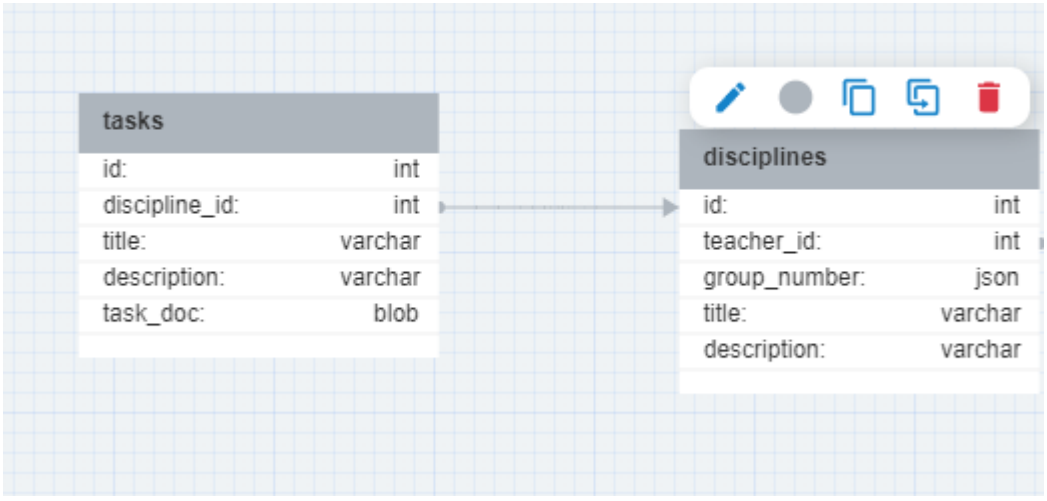


Рисунок 3.18 – Зв’язок один-до-багатьох завдань та дисциплін

Наступний зв’язок демонструє як пов’язана таблиця виконаних робіт з користувачами а також усіма завданнями. Між таблицями завдань, користувачів та виконаних завдань зв’язок типу один-до-одного.

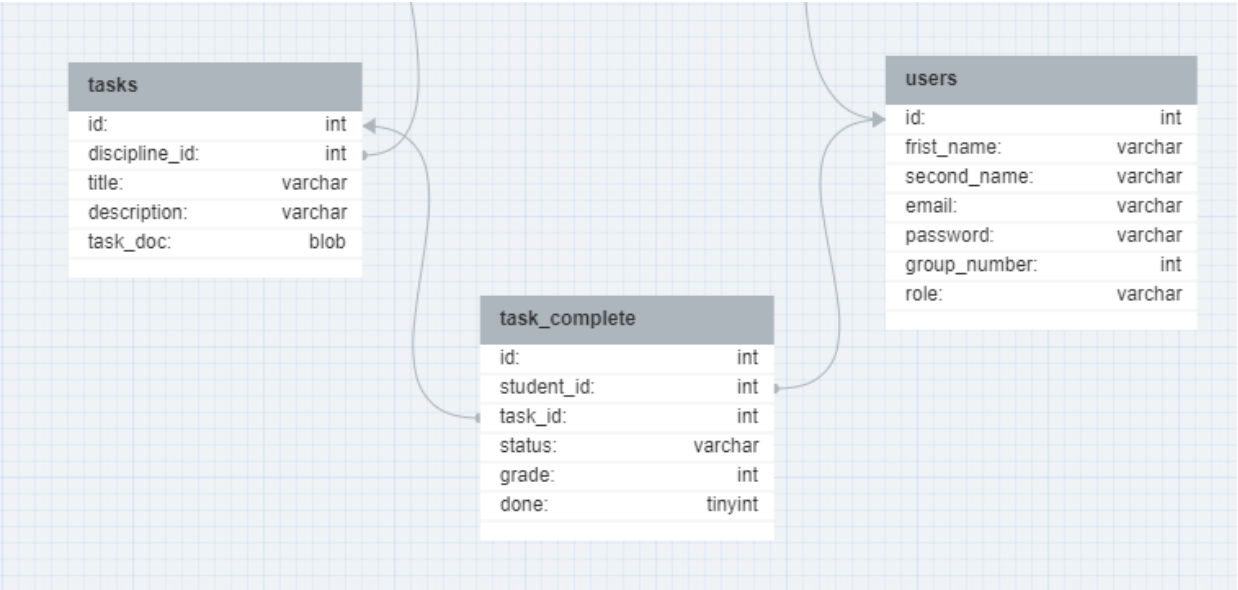


Рисунок 3.19 – Зв’язок декількох таблиць за типом один-до-одного

Routing

Архітектура роутингу в Angular включає в себе декілька ключових компонентів і концепцій. Основним компонентом є Router, який обробляє URL-адреси і вибирає компоненти для відображення. Роутер має конфігураційні правила, описані в NgModule, за допомогою методу RouterModule.forRoot(). Компоненти, що відображаються під час навігації, визначаються за допомогою RouterModule.forChild().

Для навігації використовуються посилання або методи програмної навігації, такі як router.navigate(). Angular підтримує параметризовані шляхи, що дозволяють передавати параметри в URL-адресу. Роутер також зберігає стан навігації, що дозволяє повертатись до попередніх сторінок. Існують захисні механізми, такі як

route guards, що дозволяють захистити маршрути від несанкціонованого доступу.

Angular також підтримує вкладені маршрути, що дозволяють організовувати ієрархію сторінок. Це дозволяє створювати складні структури навігації з багатьма рівнями вкладеності. Крім того, роутинг в Angular надає можливість використовувати анімацію при переходах між сторінками, що поліпшує візуальний досвід користувача.

В нашому випадку додаток матиме у своїй основі 2 роутинг модуля. Перший буде основним – він розділятиме модуль реєстрації користувача з модулем основного контенту.

```
const appRoutes: Routes = [
    {
        path: '',
        redirectTo: AppData.AppEnum.CONTENT,
        pathMatch: 'full'
    },
    {
        path: AppData.AppEnum.CONTENT,
        canActivate: [RoleGuard, AuthGuard],
        loadChildren: () =>
            import('./models/content/content.module').then(child => child.ContentModule),
    },
    {
        path: AppData.AppEnum.AUTH,
        canActivate: [AuthGuard],
        loadChildren: () =>
            import('./models/auth/auth.module').then(child => child.AuthModule),
    },
];

@NgModule({
    imports: [RouterModule.forRoot(appRoutes)],
    exports: [RouterModule],
})
export class AppRoutingModule {}
```

Рисунок 3.20 – Основний routing модуль проекту

Для початку в основному роутинг модулі AppRoutingModule оголошуємо массив доступних у додатку роутів appRoutes з типом Routes. Усередині цього масиву

ми вказуємо шляхи, при переході на які наша мережа відповідатиме підгрузкою певного модулю. Це лише частина фронт-енд, тут ми лише даємо змогу в залежності від поточного шляху відображати потрібний контент.

Параметр path допомагає нам зареєструвати назву шляху, на який може перейти користувач. Після цього маємо параметр canActivate. На цьому етапі застосовуються наші guards. Вони перевірятимуть певні умови, і в залежності від результату пропускатимуть користувача далі або ж ні. LoadChildren дозволяє нам по отриманому шляху отримати модуль, і завантажувати його у випадку переходу на конкретний роут.

Оскільки RoutingModule усе одно залишається модулем – ми повинні його зареєструвати у головному модулі проекту AppModule.

Другий модуль схожий за структурою. Він відповідає за відображення та створенням шляхів у частині з основним контентом. Основна його відмінність полягає у тому, що тепер у нас наявний початковий шлях, який помічається пустим рядком, але насправді він вже міститиме надпис з попереднього роутинг модулю ‘content’. Разом з ним завантажується компонента з контентом а також другорядні модулі.

Тепер кожний API починатиметься з content. Наприклад /content/profile, /content/tasks і т.д. В залежності від обраного шляху по такому ж самому принципу користувач отримає відповідний інтерфейс.

```
const routes: Routes = [
    {
        path: '',
        component: ContentComponent,
        children: [
            { path: '', redirectTo: AppData.AppEnum.PROFILE, pathMatch: 'full'},
            {
                path: AppData.AppEnum.PROFILE,
                loadChildren: () =>
                    import('./modules/profile/profile.module').then(child => child.ProfileModule),
                    component: ProfileComponent
            },
            {
                path: AppData.AppEnum.TASKS,
                loadChildren: () =>
                    import('./modules/tasks/tasks.module').then(child => child.TasksModule),
                    component: TasksComponent
            },
            {
                path: AppData.AppEnum.DESCIPLINES,
                loadChildren: () =>
                    import('./modules/desciplines/desciplines.module').then(child => child.DesciplinesModule),
                    component: DesciplinesComponent
            },
            {
                path: AppData.AppEnum.USERS,
                loadChildren: () =>
                    import('./modules/users/users.module').then(child => child.UsersModule),
                    component: UsersComponent
            }
        ]
    },
]

@NgModule({
    imports: [RouterModule.forChild(routes)],
    exports: [RouterModule],
})
export class ContentRoutingModule { }
```

Рисунок 3.21 – Routing модуль для контенту мережі

Цей модуль також буде підключено до іншого модуля. Уся ця система доповнюватиме роути одне-одного і забезпечить користувачеві зручну навігацію у нашій мережі а також відображуваний контент.

Висновки до розділу 3

У третьому розділі було розглянуто вхідні дані а також архітектуру та проектування інформаційної системи.

Спочатку ми визначились з основною операційною системою – Windows. Основна частина функціонування освітньої соціальної мережі здійснюватиметься через текстові дані. Під «текстовими даними» розуміємо не лише повідомлення, а й усю реєстраційну інформації про користувачів. Другорядними, але не менш важливими, даними в нашій системі є числові дані, файлові дані, таблиці.

У другому розділі було визначено архітектурні моделі нашої мережі. Ми також зазначили, що фронтова частина додатку побудована на модульній архітектурі Angular. Це дозволить нам підвищити продуктивність системи, оскільки завдяки підгрузці модулів лише під час переходу на певний route, ми зможе реалізувати принци лінивої загрузки.

Також ми детально розглянули, яку БД будемо використовувати разом з усіма її таблицями. Проаналізували дані, наявні у кожній з таблиць, а також типи зв’язки між ними.

Завдяки грамотно-побудованій архітектурі ми зможемо забезпечити безболісне розширення функціоналу мережі у майбутньому і зберегти ресурси, які відображатимуться на продуктивності взаємодії застосунку та користувачів.

4. ПРОГРАМНА РЕАЛІЗАЦІЯ ОСВІТНЬОЇ СОЦІАЛЬНОЇ МЕРЕЖІ

4.1 Підключення бази даних

Оскільки процес проектування бази даних обговорено у минулому розділі, пропоную одразу перейти до її підключення.

Для взаємодії з базою даних скористаємося MVC патерном у нашому підпроекті під назвою server.

У файлі з конфігом форматом json збережемо необхідні нам дані для підключення. Вони матимуть наступний вигляд.

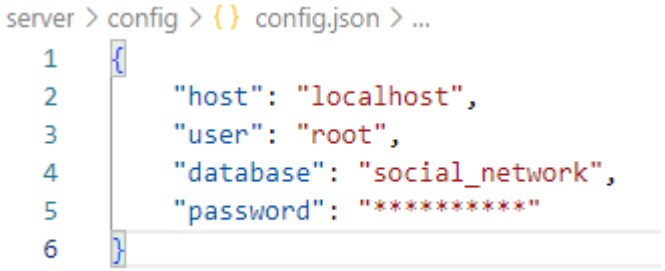


Рисунок 4.1 – Конфіг підключення до БД

У файлі database.js отримаємо ці дані, створимо під них змінні і пул підключень. Для цього нам буде необхідний модуль під назвою mysql2. Пули дозволяють нам зменшити час на підключення до серверу MySQL, завдяки повторному використанню підключень.

server > util > JS database.js > ...

```
1   const mysql = require('mysql2');
2
3   const {user, host, password, database} = require('../config/config.json');
4
5   const pool = mysql.createPool({
6       host,
7       user,
8       password,
9       database
10  });
11
12  module.exports = pool.promise();
```

Рисунок 4.2 – Пул підключення до БД

Після підключення і створення пулу ми можемо використовувати усі можливості нашої БД. Як приклад буде продемонстровано основні CRUD операції для роботи з таблицею усіх користувачів.

Експортуємо наший пул з базою даних та створюємо клас користувачів, який прийматиме усі необхідні для бази даних поля. Також під кожну взаємодію прописуємо метод, а саме: отримання усіх користувачів, додавання нового користувача, оновлення даних про користувача а також видалення користувача за його айді.

Основний принцип такий: у об’єкта бази даних викликаємо метод execute(), який на вході приймає SQL запит.

server > models > JS users.js > db

```
const db = require('../util/database');

module.exports = class Users {
    constructor(id, first_name, second_name, email, password, group_number, role, avatar) {
        this.id = id;
        this.first_name = first_name;
        this.second_name = second_name;
        this.email = email;
        this.password = password;
        this.group_number = group_number;
        this.role = role;
        this.avatar = avatar;
    }

    static getAll() {
        return db.execute('SELECT * FROM users');
    }

    static post(user) {
        return db.execute('INSERT INTO users (first_name, second_name, email, password, group_number, role, avatar) VALUES (?)',
        [user.first_name, user.second_name, user.email, user.password, user.group_number, user.role, user.avatar]);
    }

    static update(id, user) {
        return db.execute('UPDATE users SET user = ? WHERE id = ?', [user, id]);
    }

    static delete(id) {
        return db.execute('DELETE FROM users WHERE id = ?', [id]);
    }
}
```

Рисунок 4.3 – Створення методів взаємодії з БД

Наступним етапом ми переходимо до папки з моделями та імпортуємо наш клас User.

Створюємо на кожну CRUD операцію метод у нашому контролері, і вже зі створених методів у папці з моделями використовуємо їх відповідно. Таким чином ми розділяємо бізнес-логіку.

Оскільки запити до бази даних це асинхронні події – використовуємо async/await стрілочні функції задля очікування результату від запиту. Там, де метод з моделей очікує вхідних параметрів, ми передаємо тіло запиту req.body аби надалі «розпакувати» його і отримати необхідну інформацію.

Після отриманого результату відправляємо запиту статус 201 а також у форматі json надсилаємо отримані дані.

Усі операції виконуються усередині блоку try/catch задля перехоплення і опрацювання можливої помилки.

```
exports.getAllRoutes = async (req, res, next) => {
    try {
        const [allRoutes] = await Users.getAll();
        res.status(200).json(allRoutes)
    } catch(err) {
        if(!err.statusCode) err.statusCode = 500;
        next(err);
    }
};

exports.postUser = async (req, res, next) => {
    try {
        const userResponse = await Users.post(req.body);
        res.status(201).json(userResponse);
    } catch(err) {
        if(!err.statusCode) err.statusCode = 500;
        next(err);
    }
}

exports.putUser = async (req, res, next) => {
    try {
        const userResponse = await Users.update(req.body);
        res.status(201).json(userResponse);
    } catch(err) {
        if(!err.statusCode) statusCode = 500;
        next(err);
    }
}
```

Рисунок 4.4 – Виклик функцій з контролеру

Нарешті ми можемо прописати шляхи для нашого Route, а також вказати які функції будуть їх опрацьовувати. Налаштований router обов’язково експортуємо для реєстрації у глобальному середовищі додатку з усіма можливими endpoints а також middlewears.

```
const express = require('express');

const usersController = require('../controller/users');

const router = express.Router();

router.get('/', usersController.getAllRoutes);

router.post('/', usersController.postUser);

router.put('/', usersController.putUser);

module.exports = router;
```

Рисунок 4.5 – Підв’язка функцій з контролеру до http запиту

4.2 Створення користувачів

Наступним важливим етапом є додання користувачів до бази даних, їх розподіл на адміністраторів, викладачів та студентів. Відповідно до кожної ролі буде надано певний функціонал, про який вже було згадано у розділах вище. З цією метою у системі існуватиме два етапи: аутентифікація та авторизація.

Аутентифікація – це процес перевірки та підтвердження ідентичності користувача, пристрою або системи з метою надання доступу до певних ресурсів, інформації або послуг. Це важлива складова безпеки, яка дозволяє перевірити, що особа або сутність, яка намагається отримати доступ, дійсно є тим, за кого себе видає. Вона базується на знаннях, володінні або характеристиках, що є унікальними для користувача, пристрою або системи.

Авторизація – це процес надання прав доступу користувачеві, пристрою або системі після успішної аутентифікації. У процесі авторизації визначається, до яких ресурсів, інформації або послуг має право доступу конкретний користувач. Після того, як особа або сутність пройшла аутентифікацію та підтвердила свою ідентичність, авторизація визначає, які дії або операції може виконати цей користувач. Вона

встановлює права і обмеження доступу, які визначають, які функції, ресурси або дані є доступними для використання.

```
<mat-card>
    <mat-card-title>Login</mat-card-title>
    <mat-card-content>
        <form [formGroup]="form" (ngSubmit)="logIn()">
            <p>
                <mat-form-field>
                    <input type="email" matInput placeholder="Email" formControlName="email">
                </mat-form-field>
            </p>
            <p>
                <mat-form-field>
                    <input type="password" matInput placeholder="Password" formControlName="password">
                </mat-form-field>
            </p>
            <div class="button">
                <button type="submit" color="primary" mat-raised-button>Login</button>
            </div>
            <div class="button reg">
                <button mat-raised-button>
                    <a routerLink="/auth/register" class="register" (click)="logIn()">Registrate new one</a>
                </button>
            </div>
        </form>
    </mat-card-content>
</mat-card>
```

Рисунок 4.6 – Форма авторизації html

Як бачимо, дизайн форми виконано з використанням UI бібліотеки Angular Material. Усього форма авторизації матиме два поля – назва пошти та пароль від аккаунту. Кнопка логіну має тип submit. Якщо користувач хоче зареєструвати новий аккаунт – по натисканню на кнопку його буде переведено на роут /auth/register з формою реєстрації.

Для логіну і усіх інших форм у нашій мережі ми будемо користуватись широким функціоналом Angular, який надає нам такий зручний інструмент, як реактивна форма. Для цього визначається сама форма, а також параметром formControlName імена полів форми, з яких ми потім отримуватимемо інформацію.

Саму форму необхідно оголосити всередині компоненти, тип форми:

FormGroup. При оголошенні також ми задаємо наявні поля, їхнє початкове значення а також можемо додати валідатори, аби виконувати перевірку введених даних не після відправлення запиту до бази даних, а ще до цього моменту.

```
public readonly form = new FormGroup({
  email: new FormControl(null, [
    Validators.required,
    Validators.email
  ]),
  password: new FormControl(null, [
    Validators.required
  ])
})
```

Рисунок 4.7 – Оголошення форми для авторизації з валідацією полів

У разі, якщо користувачем при авторизації були введені правильні дані, створиться JWT токен. Він відіграє важливу роль у нашій мережі, оскільки економить наші ресурси і дозволяє досить простим чином отримувати інформацію про поточного користувача: його роль, ПІБ, номер групи та всі інші наявні дані.

Наявність JWT токену також забезпечить нам перехід до основного контенту мережі, оскільки створені guards не допустять переходу користувача до інших роутів, окрім сторінок авторизації. Але давайте по порядку.

Для роботи з JWT токеном і express нам знадобляться наступні пакети:

```
"dependencies": {
    "body-parser": "^1.20.2",
    "cors": "^2.8.5",
    "express": "^4.18.2",
    "jsonwebtoken": "^9.0.0",
    "mysql2": "^3.3.1",
    "nodemon": "^2.0.22"
  }
```

Спочатку ми створюємо додаток Express і визначаємо секретний ключ, який буде використовуватись для кодування і декодування даних користувача всередині JWT токену.

Ми не будемо для збереження токену використовувати базу даних, але тим не менш концепції були б схожими, якщо ми б писали правильний сервер з такою підтримкою.

Також нам необхідно створити маршрути для аутентифікації користувача. Приклад наведено нижче.

```
const cors = require('cors');
const bodyParser = require('body-parser');
const jwt = require('jsonwebtoken');
const express = require('express');

const app = express();
app.use(cors());
app.use(bodyParser.json());

const JWT_Secret = 'secret_key_for_diploma';

var user = { };

app.post('/api/authenticate', (req, res) => {
    if (req.body) {
      var user = req.body;
      console.log(user)
      if (user == req.body) {
        var token = jwt.sign(user, JWT_Secret);
        res.status(200).send({
          signed_user: user,
          token: token
        });
      } else {
        res.status(403).send({
          errorMessage: 'Authorisation required!'
        });
      }
    } else {
      res.status(403).send({
        errorMessage: 'Please provide email and password'
      });
    }
```

Рисунок 4.8 – Створення та запис JWT токену

Далі ми пишемо код, який виконує POST-запит до серверу з реєстраційними даними користувача. Тут ми робимо запит до API – якщо він успішний, ми зберігаємо токен в LocalStorage і перенаправляємо користувача на сторінку профіля.

```
@Injectable({
    providedIn: 'root'
})
export class JWTService {
  public uri = 'http://localhost:5000/api';
  public token;

  constructor(
    private http: HttpClient,
    private router: Router
) { }

  login(email: string, password: string) {
    this.http.post(this.uri + '/authenticate', {email: email,password: password})
    .subscribe((resp: any) => {

      this.router.navigate(['profile']);
      localStorage.setItem('auth_token', resp.token);

      })
    }

  logout() {
      localStorage.removeItem('token');
    }
    public get logIn(): boolean {
      return (localStorage.getItem('token') !== null);
    }
}
```

Рисунок 4.9 – Управління токеном при login та logout

Тепер завдяки токену ми зможемо отримувати інформацію про користувача з будь-якої компоненти, для цього нам потрібно буде лише зчитати токен.

4.3 Реалізація та застосування соціального графу

Як вже було зазначено вище, соціальний граф – модель, яка відображає взаємозв'язки між людьми в соціальній мережі. Він складається з вузлів (людей) і зв'язків (стосунків) між ними. Соціальний граф може мати велике значення в різних сферах, оскільки він надає важливі дані про соціальні зв'язки і взаємодії між людьми.

У даній соціальній мережі завдяки соціальному графу ми реалізуємо рекомендаційний список користувачів, виходячи з критерію схожості дисциплін, що вивчаються, а також середнього балу студента.

Для початку необхідно встановити необхідну залежність-бібліотеку для відображення графа. Це буде одна з популярних бібліотек `ngx-graph`. Встановимо за допомогою npm:

npm i @swimlane/ngx-graph

Для його відображення та взаємодії створимо окрему компоненту та реалізуємо шаблон графу.

```
<ngx-graph [nodes]="graphNodes" [links]="graphLinks">
    <ng-template #nodeTemplate let-node>
      <svg:g class="node">
        <svg:rect [attr.width]="node.width" [attr.height]="node.height" [attr.fill]="node.color"></svg:rect>
        <svg:text>{{ node.label }}</svg:text>
      </svg:g>
    </ng-template>
  </ngx-graph>
```

Рисунок 4.10 – Шаблон соціального графу

В основному компоненті створимо методи отримання та становлення інформації про граф. Уся інформація отримується завдяки http запитам, які повертають нам реактивний об'єкт, на який треба підписатись. Завдяки методу subscribe() розгортаємо об'єкт data.

```
@Component({
    selector: 'app-graph',
    templateUrl: './social-graph.component.html',
    styleUrls: ['./social-graph.component.scss']
})
export class SocialGraphComponent {
    graphNodes = [];
    graphLinks = [];

  constructor(
    private http: HttpClient,
    private readonly _usersService: UsersService
  ) { }

  ngOnInit(): void {
    this.getGraphData();
  }

  getGraphData() {
    this.http.get<any>('/api/graph').subscribe(data => {
      this.graphNodes = data.nodes;
      this.graphLinks = data.links;
    });
  }

  saveGraphData() {
    const data = { nodes: this.graphNodes, links: this.graphLinks };
    this.http.post<any>('/api/graph', data).subscribe(response => {
      console.log(response.message);
    });
  }
}
```

Рисунок 4.11 – Компонент соціального графа

Тепер нам необхідно налаштувати API на NodeJS для того, щоб зареєструвати шляхи, по яким будуть посилатись запити а також зберігання отриманих даних на сервері.

Маршрут GET /api/graph повертає дані графа з бази даних або іншого джерела. Зараз дані графа хардкодовані, але ви можете модифікувати код, щоб отримувати дані з реального джерела, наприклад, бази даних.

Маршрут POST /api/graph приймає дані графа з клієнта та зберігає їх до бази даних або іншого джерела.

```
export class SocialGraphComponent {
    graphNodes = [];
    graphLinks = [];

  constructor(
    private http: HttpClient,
    private readonly _usersService: UsersService
  ) { }

  ngOnInit(): void {
    this.getGraphData();
  }

  getGraphData() {
    this.http.get<any>('/api/graph').subscribe(data => {
      this.graphNodes = data.nodes;
      this.graphLinks = data.links;
    });
  }

  saveGraphData() {
    const data = { nodes: this.graphNodes, links: this.graphLinks };
    this.http.post<any>('/api/graph', data).subscribe(response => {
      console.log(response.message);
    });
  }
}
```

Рисунок 4.12 – API routes для соціального графа

4.4 Користувацький функціонал

Після вдалого логіну у свій аккаунт, спрацює authGuard, який перевірятиме наявність токену, а тому користувача буде переслано на сторінку з основним контентом а точніше на його профіль.

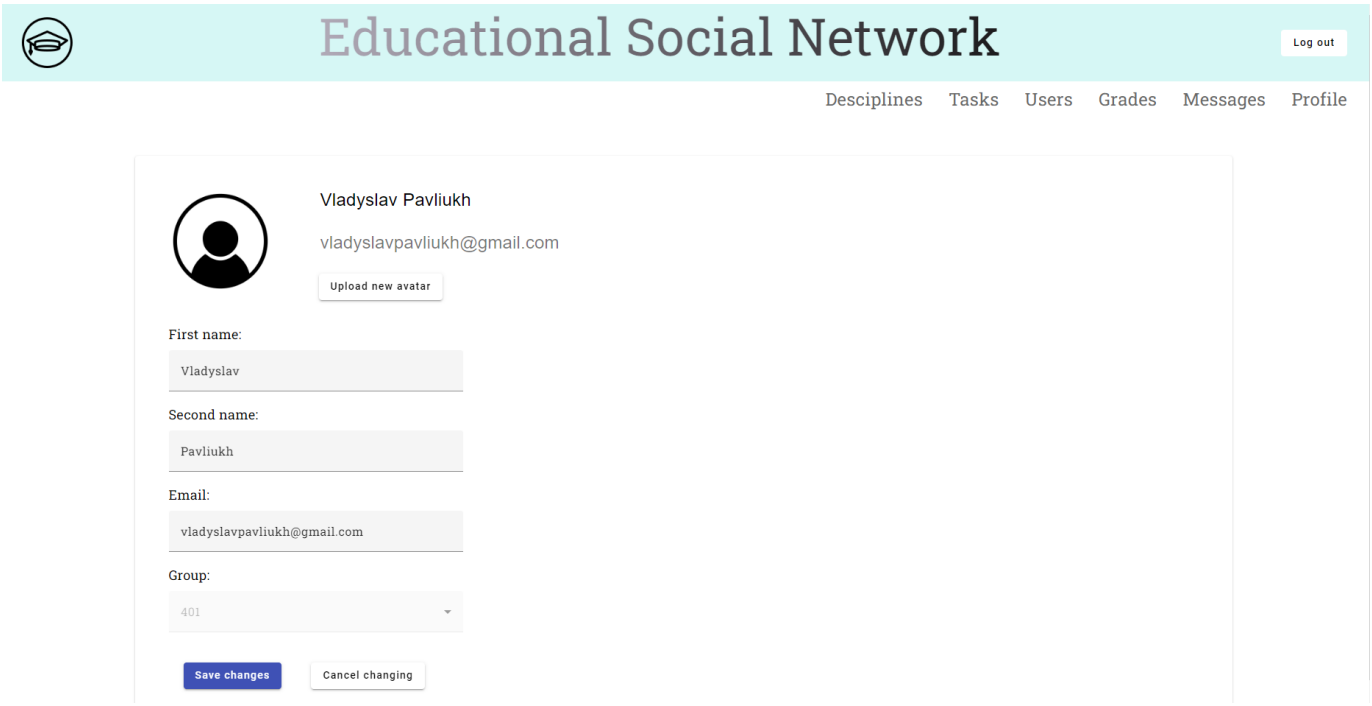


Рисунок 4.13 – Профіль користувача

Для користувачів з ролями admin та teacher поле group буде відсутнє.

Взагалі, увесь розподіл функціоналу мережі, в залежності від його ролі, буде виконано через <ng-template></ng-template> у Angular. Тобто у сервісі з користувачами ми створюємо метод, який читає зі створеного токену інформацію про користувача і повертає лише його роль. В залежності від ролі ми створюватимемо змінну, від значення якої буде застосовуватись той чи інший інтерфейс.

Наступною буде сторінка з користувачами. Ця компонента реалізована наступним чином: для пошуку потрібного користувача необхідно обрати його роль а також ввести ім’я або фамілію, у разі знаходження цього користувача буде додано до загального списку.

Якщо користувачем є студент, то йому відображаються його однокурсники та користувачі, яких він знайде. Якщо користувачем є викладач – йому відображатимуться його студенти а також знайдені користувачі. У випадку адміністратора – він зможе, на відміну від двох попередніх типів користувача, не лише

додавати до друзів та писати у особисті повідомлення, а й видаляти користувача. В такому випадку користувача буде видалено із бази даних, усі його дані буде втрачено.

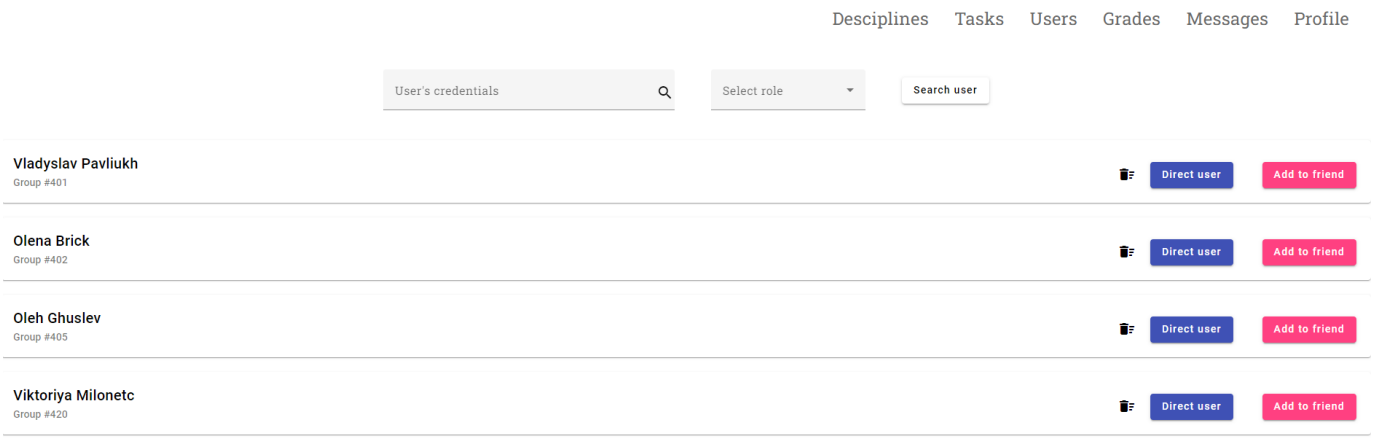
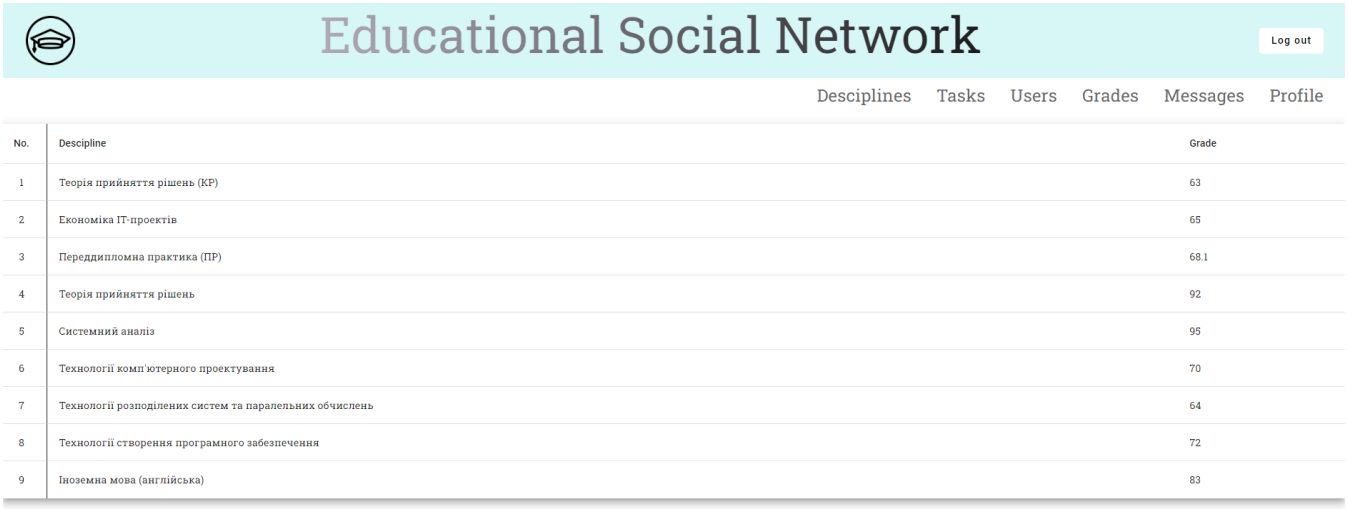


Рисунок 4.14 – Сторінка користувачів системи

Якщо поточний користувач має роль студента або викладача, то йому буде відображено вкладку з оцінками.



No.	Discipline	Grade
1	Теорія прийняття рішень (КР)	63
2	Економіка IT-проектів	65
3	Переддипломна практика (ПР)	68.1
4	Теорія прийняття рішень	92
5	Системний аналіз	95
6	Технології комп'ютерного проектування	70
7	Технології розподілених систем та паралельних обчислень	64
8	Технології створення програмного забезпечення	72
9	Іноземна мова (англійська)	83

Рисунок 4.15 – Сторінка оцінок студента

У випадку вчителя – буде декілька таблиць, по кожній групі, у випадку студента – одна таблиця з полями дисципліна та оцінка.

Для вчителів сторінка з дисциплінами виглядає наступним чином. Вчитель має можливість як створити нову дисципліну, вказавши назву, опис та номер групи, або ж взаємодіяти зі вже створеними (редагувати та видаляти). Для студентів такого функціоналу не буде, для них доступний лише перелік дисциплін, які будуть представлені схожим чином у вигляді посилань. При натисканні на дисципліну користувача буде переносити на наступну сторінку: сторінку з завданнями, де вони зможуть отримувати домашнє завдання а також завантажувати свою виконану роботу.

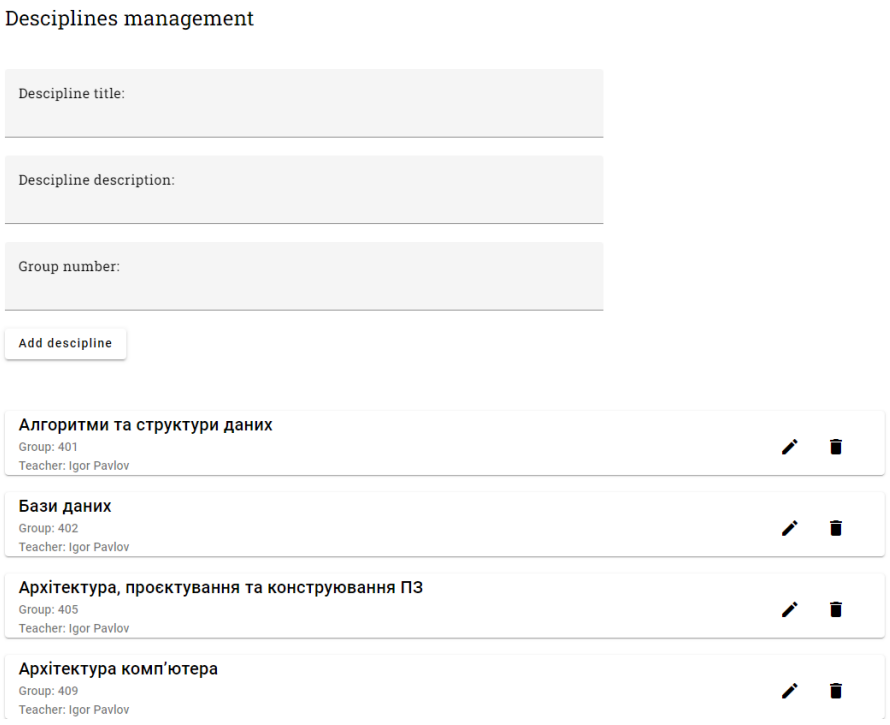


Рисунок 4.16 – Сторінка з дисциплінами

Сторінка з завданнями у студентів буде у вигляді випадаючого списку. Студент матиме перелік предметів, при кліку на яких з'являться усі доступні домашні завдання. Клік на завдання перенесе користувача на нову компоненту, де він зможе

завантажити файл з завданням а також відправити власне.

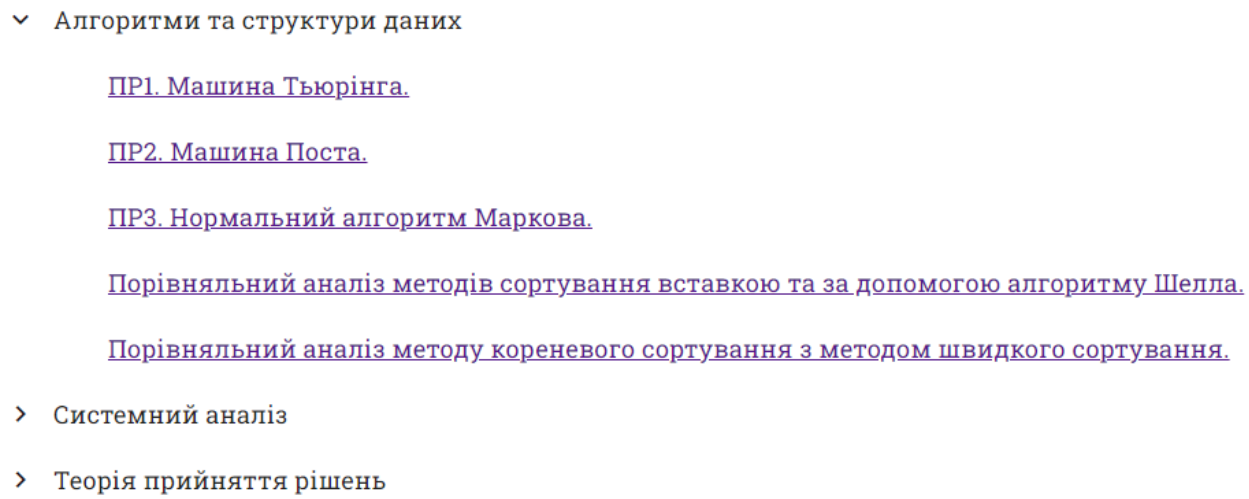


Рисунок 4.17 – Наявні завдання у студентів

Кожне завдання представлене у вигляді посилання, при створені елементу ми додаємо айді завдання до шляху і він має наступний вигляд:

```
{
                path: `${AppData.AppEnum.TASKS}/:id`,
                loadChildren: () =>
                    import('./modules/tasks/tasks.module').then(child =>
child.TasksModule),
                    component: TaskPageComponent
},
```

На відкритій сторінці студент зможе побачити доданий викладачем опис завдання, прикріплений файл а також місце для завантаження власної виконаної роботи.

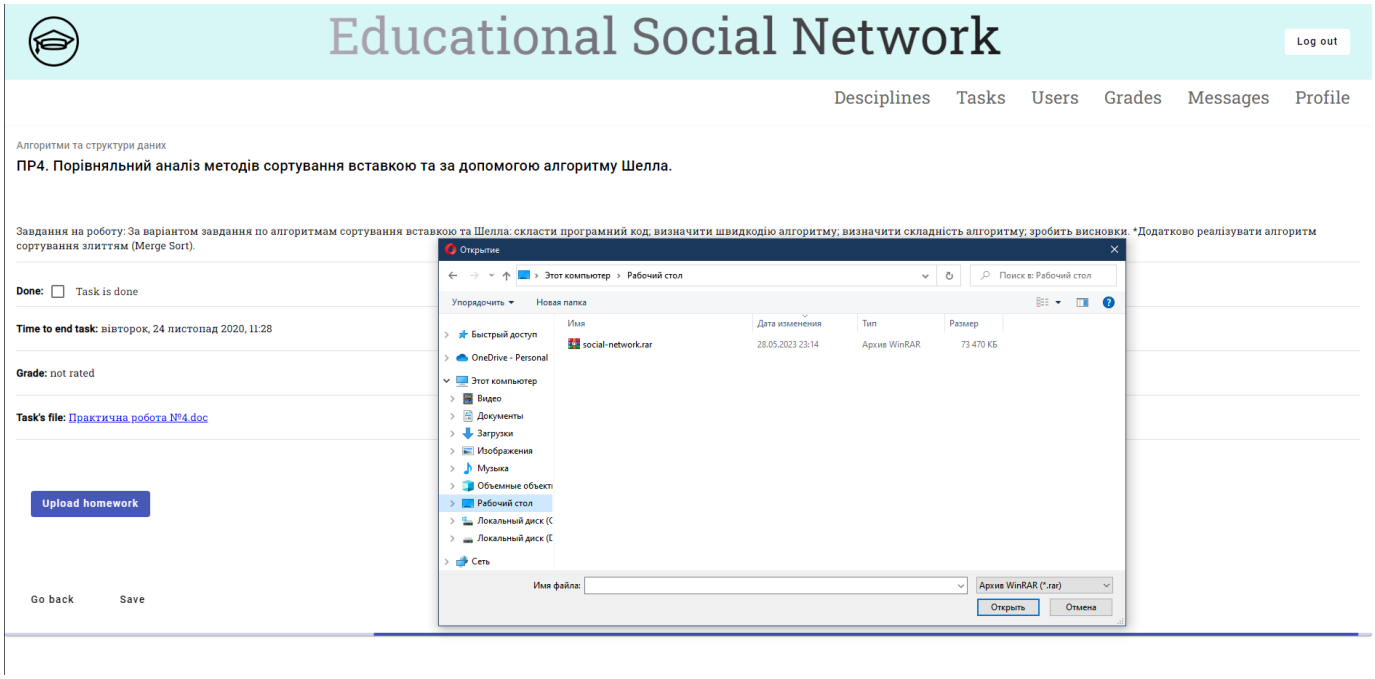


Рисунок 4.18 – Завантаження виконаного завдання студентом

Як видно з рисунку, студенту доступна інформація про термін здачі роботи, оцінювання роботи а також файл з завданням, який завантажив викладач. Сам студент може завантажувати в мережу лише файли збережені в архіві. Їх доволі зручно зберігати, незалежно від кількості файлів, які необхідні для звіту.

Останньою сторінкою нашої мережі залишилась сторінка з чатом користувача. Тут зберігається інформація про усіх співрозмовників поточного користувача, при клік на яких відкривається історія повідомлень між користувачами.

Процес реалізовано завдяки запитам до бази даних, де містяться усі повідомлення у таблиця «повідомлення від» і «повідомлення до». Нам залишається лише обрати ті записи в таблиці, де «повідомлення від» збігаються з id поточного користувача, усіх з «повідомлення до» записати в історію співрозмовників. А далі запит буде здійснюватись після кліку на конкретного користувача, ми будемо просто проходитись по записам таблиці з поточним та обраним користувачами та відображати їхні повідомлення.

Рисунок 4.19 – Сторінка з чатами користувачів

Висновки до розділу 4

В даному розділі було описано розробку та програмну реалізацію освітньої соціальної мережі для викладачів та студентів. Було розглянуто інтерфейс функціоналу в залежності від різних ролей користувача, а саме сторінки: реєстрації та авторизації, профілю користувача, журналу оцінок для студентів, списку користувачів та одногрупників, дисциплін, завдань та чату між усіма користувачами мережі.

Для користувачів було застосовано соціальний граф. Завдяки результатам графу можна визначити рекомендаційний лист з потенційними друзями в мережі. Підбірка складається на основі предметів, що вивчаються а також середньому балу студентів.

ВИСНОВКИ

В умовах наростання проблем які несуть глобальну загрозу можливості сталого розвитку деяким країнам, освіта та наука бере на себе завдання виховати фахівців, які зможуть впоратися з цими проблемами. Наука виконує тепер місію збереження балансу між природою та людиною. Тепер головною місією освіти стає не лише передача знань, але й розкриття творчого потенціалу кожної людини.

Проведене дослідження дозволяє зробити наступні висновки.

Аналіз сучасної системи освіти показав, що в умовах недавніх років освітній процес супроводжується і цілком будується навколо дистанційної освіти. Вона є необхідним елементом сучасного світу, що надає унікальні можливості для навчання та отримання освіти за допомогою електронних засобів комунікації. Завдяки глобалізації та зростанню доступу до Інтернету, дистанційна освіта стала доступною для усіх, незалежно від їх географічного розташування.

Установлено, що освітній процес у нинішніх умовах можна покращити шляхом поєднання процесів комунікації та взаємодії викладачів та студентів. З цією метою було розроблено соціальну мережу, яка поєднала у собі основний функціонал конкретного вищого навчального закладу, і в разі подій минулої осені та зими надасть можливість як студентам так і викладачам виконувати свої обов’язки у більш зручних умовах.

Серед інструментальних засобів розробки освітньої соціальної мережі було обрано мову програмування TypeScript, фреймворк Angular, який вміщує у собі великий обсяг функціоналу, для backend частини було обрано NodeJS, база даних – MYSQL. Для роботи з базою даних, як і з більшою частиною сутностей в програмі, були завантажені додаткові пакети завдяки npm. У якості інтегрованого середовища розробки було обрано Visual Studio.

У результаті проведеного дослідження було здійснено розробку та програмну

реалізацію освітньої соціальної мережі для викладачів та студентів, яка стане замінником добре нам відомого Moodle. Найбільше можливостей у мережі матиме адміністратор, він матиме доступ до управління користувачами. Викладачі зможуть створювати навчальні дисципліни під кожну групу, а також завантажувати завдання і оцінювати здані роботи. Найменше прав доступу матимуть студенти, для них, як і для всіх, буде доступний пошук і комунікація з усіма користувачами мережі, а також вони зможуть переглядати свої доступні дисципліни та здавати вчасно роботи.

У спеціальному розділі було проаналізовано санітарно-гігієнічні вимоги приміщення для роботи з комп’ютерною технікою, розглянуто вимоги до усіх сторін за предметним розділом Охорони праці, визначено режими відпочинку та праці на підприємстві.

Поставлені завданні виконані у повному обсязі. У разі наявності бажання покращення інтерфейсу чи розширення функціоналу не має виникнути проблем через використання UI бібліотеки Angular Material та принципів ООП, які забезпечують «безболісне» розширення програми.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Education at a Glance 2011. OECD indicators. – Paris: OECD Publishing, 2011. P. 232. URL: https://www.oecd.org/education/skills-beyond-school/48631582.pdf (дата звернення 03.05.2023).

2. Education at a Glance 2012. OECD indicators. – Paris: OECD Publishing. 2012. P. 360. URL: https://www.oecd-ilibrary.org/docserver/eag-2012-en.pdf?expires=1687128889&id=id&accname=guest&checksum=0B708DF7F3F3F58EB51FDC4BD5A7568D (дата звернення 03.05.2023).

3. Андрущенко, В. Основні тенденції розвитку вищої освіти України на рубежі століть (спроба прогностичного аналізу). 2001. № 1. С.11-17. (дата звернення: 04.05.2023).

4. Вища освіта в Україні: порядок денний для реформ / за заг. ред. Ніколаєва Є.Б. – К. : Представництво Фонду Конрада Аденауера в Україні. 2017. 61 с. (дата звернення: 04.05.2023).

5. Болюбаш Я. Я. Організація навчального процесу у вищих закладах освіти: Навч. Посібник для слухачів закладів підвищення кваліфікації системи вищої освіти. К.: ВВП «КОМПАС». 1997. 64 с. (дата звернення: 05.05.2023).

6. Вища освіта України: Теоретичний та науково-методичний часопис. 2015. 29 с. URL: https://ihed.org.ua/wp-content/uploads/2015/10/jurn_VishaOsvitaUkraini_IntgracOsvitiNauki_IVO2015.pdf. (дата звернення: 05.05.2023).

7. Вища освіта України: реалії сучасного розвитку / С.О. Сисоєва, Н.Г. Батечко / Міністерство освіти і науки, молоді та спорту України, Київський університет імені Бориса Грінченка, Національний університет біоресурсів і природокористування України. К.: ВД ЕКМО, 2011. 368 с. (дата звернення: 05.05.2023).

8. Антонова О. С., Колосюк А. А. Роль социальных сетей в системе образования / Финансовая система Украины: проблемы и перспективы развития в условиях трансформации социально-экономических отношений: материалы междунар. научно-практ. конф. Севастополь: ДІАПІ. 2013. С. 211–213. (дата звернення: 06.05.2023).

9. Кадемія М. Ю. Соціальні сервіси веб 2.0 в освітній діяльності / Матеріали методологічного семінару кафедри інформаційних технологій в освіті 2010-2011 р. Вінницького державного педагогічного університету імені Михайла Коцюбинського. URL: http://ito.vspu.net/SAIT/inst_kaf/kafedru/matem_fizuka_tex_osv/WWW/metod_seminar/2008/kademiya/kademiya_2010-2011.htm (дата звернення 07.05.2023).

10. Бершадська Л.А., Біккулов А.С., Болгова Є.В., Чугунов А.В., Якушев А.В. Соціометричного дослідження в соціальних мережах як інструментарій соціології і політології / Сучасні проблеми науки та освіти. 2012. № 4. URL: http://www.scienceeducation.ru/ru/article/view?id=6901 (дата звернення: 09.05.2023).

11. Положення про дистанційне навчання [Архівовано 17 червня 2019 у Wayback Machine.], Наказ МОН № 40 від 21 січня 2004 року. (дата звернення: 10.05.2023).

12. Історія національної освіти і педагогічної думки в Україні : Навч. посіб. / Л. А. Медвідь. - К. : Вікар, 2003. – 335 с. - (Вища освіта XXI ст.). (дата звернення: 10.05.2023).

13. Курбатов С.В. Дистанційна освіта як складова діалогу влади і громадськості у забезпеченні права на освіту / Діалог суспільства і влади: правові форми, виклики, перспективи. К.: Інститут законодавства ВР України. 2012. С. 79-87. (дата звернення: 11.05.2023).

14. Newman M. Networks: An Introduction. Oxford University Press. 2010. P. 784. URL: https://math.bme.hu/~gabor/oktatas/SztoM/Newman_Networks.pdf. (дата звернення: 12.05.2023).

15. C. McCarthy. Facebook: One Social Graph to Rule Them All? CBS Interactive Inc., 2010. URL: https://www.cbsnews.com/stories/2010/04/21/tech/main6418458.shtml. (дата звернення: 12.05.2023).

16. T. Opsahl, F. Agneessens, J. Skvoretz. Social Networks. 2010. URL: https://toreopsahl.com/2010/04/21/article-node-centrality-in-weighted-networks-generalizing-degree-and-shortest-paths/ . (дата звернення: 13.05.2023).

17. M. P. Zillman. Online Social Networks (англ.). Virtual Private Library. 2012. URL: http://whitepapers.virtualprivatelibrary.net/Online%20Social%20Networks.pdf. (дата звернення: 13.05.2023).

18. Батура Т. В. Методы анализа компьютерных социальных сетей. 2012. Т. 10. Вып. 4. С. 13–28. URL: http://lib.nsu.ru:8080/jspui/bitstream/nsu/250/1/02.pdf. (дата звернення: 15.05.2023).

19. Cross R. Knowing what we know: supporting knowledge creating and sharing in social networks / R. Cross, A. Parker, L. Prusak, S.P. Borgatti // Organizational Dynamics. 2001. Vol. 30. № 2. P. 100–120. URL: https://www.researchgate.net/publication/230557500_Knowing_What_We_Know_Supporting_Knowledge_Creation_and_Sharing_in_Social_Networks. (дата звернення: 15.05.2023).

20. Erdos P., Renyi A. On the evolution of random graphs / Publ. Math. Inst. Hungar. Acad. Sci. 1960. V. 5. P. 17–61. URL: https://www.jstor.org/stable/1999405. (дата звернення: 16.05.2023).

21. B. R. Holland. Enabling Open Source Intelligence (OSINT) in private social networks: Masters's dessertation. — Iowa State University, Ames, Iowa. 2012. URL: https://dr.lib.iastate.edu/entities/publication/541e8e27-b8f9-4946-9cb5-a00a72fad03b. (дата звернення: 18.05.2023).

22. Бреер В. В. Стохастические модели социальных сетей / Управление большими системами. 2009. Вып.27. С. 169–204. URL:

https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&ved=2ahUKEwi2v7SL9c3_AhWNjYsKHRXXAsUQFnoECBQQAQ&url=http%3A%2F%2Fubs.mtas.ru%2Fupload%2Flibrary%2FUBS2713.pdf&usg=AOvVaw23QWImELwiygWGcVkSmFQz&opi=89978449. (дата звернення: 18.05.2023).

23. Демків О. Б. Розвиток та основні напрямки мережевого аналізу / Методологія, теорія та практика соціологічного аналізу сучасного суспільства: Збірник наукових праць. 2003. С. 161–166. URL: http://sociology-lnu.org.ua/resursy.files/Demkiv_stattja_5.htm (дата звернення: 19.05.2023).

24. Gechev M., Switching to Angular 2 / Gechev M. - Packt Publishing. 2016. P. 254. URL: https://dl.ebooksworld.ir/motoman/Packt.Switching.to.Angular.2.www.EBooksWorld.ir.pdf. (дата звернення: 20.05.2023).

25. Документація JavaScript,Typescript [Електронний ресурс]. 2020. режим доступу: http://www.w3.org/. (дата звернення: 21.05.2023).

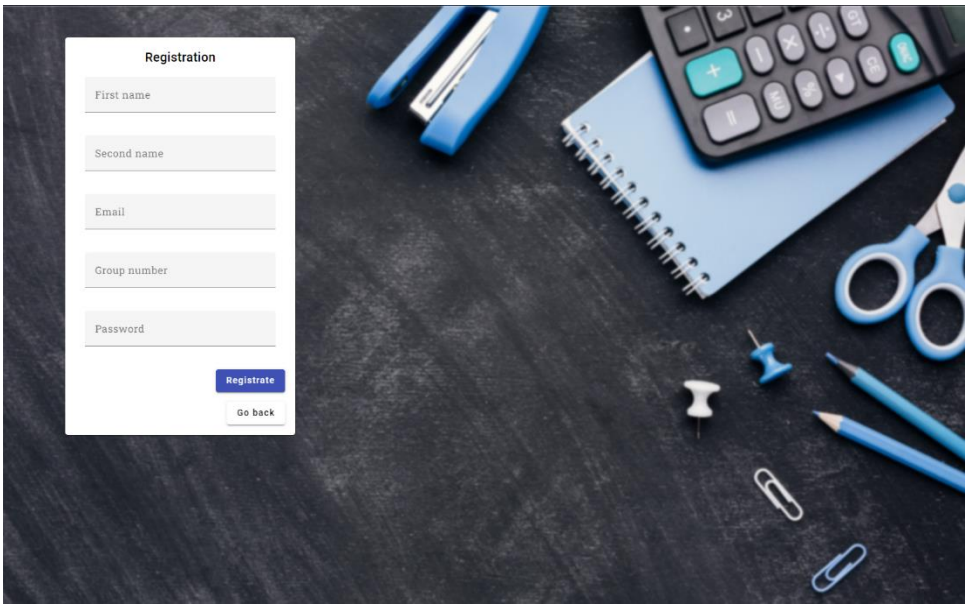
26. Schwartz B. High Performance MySQL: Optimization, Backups, and Replication / B. Schwartz, P. Zaitsev, V. Tkachenko. 2012. P. 826. (дата звернення: 23.05.2023).

27. "Angular, version 2: proprioception-reinforcement". blogspot.com. September 14, 2016. Archived from the original on 2017-03-12. Retrieved 2017-03-18. (дата звернення: 23.05.2023).

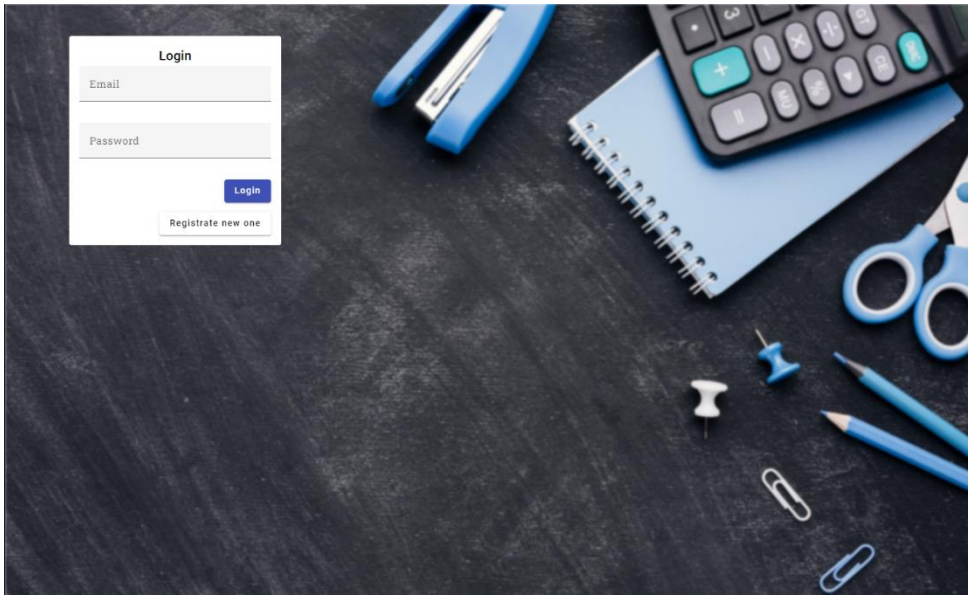
ДОДАТОК А

Форми для реєстрації та входу

Форма для реєстрації:



Форма для логіну:



ДОДАТОК Б

Інтерфейси сутностей програми

Інтерфейс користувача

```
export interface User {
            id?: number;
            first_name: string;
            second_name: string;
            email: string;
            password: string;
            group_number: number;
            role: string;
            avatar?: File
        }
```

Інтерфейс дисципліни

```
export namespace Descipline {
        export interface Descipline {
            id?: number,
            teacher_id: number,
            group_number: number,
            title: string,
            description: string
        }
    }
```

Інтерфейс завдання

```
export namespace Task {
        export interface Task {
            id?: number,
            title: string,
            description: string,
            done: boolean,
            grade: number,
            student_id: number
        }
    }
```

Інтерфейс навігації

```
export interface INav {
        title: string,
        route: string
    }
```

Інтерфейс оцінки

```
 export namespace Grades {
        export interface Grade {
            id?: number,
            student_id: number,
            completed_task_id: number
        }
    }
```

ДОДАТОК В

Guard авторизації

```
@Injectable({
  providedIn: 'root',
})
export class AuthGuard implements CanActivate {
  constructor(
    private readonly _authService: AuthService,
    private readonly _profileService: ProfileService,
    private readonly _router: Router
  ) {}

  public canActivate(
    route: ActivatedRouteSnapshot,
    state: RouterStateSnapshot
  ):
    | boolean
    | UrlTree
    | Observable<boolean | UrlTree>
    | Promise<boolean | UrlTree> {
    if (
      state.url.startsWith(`/${AppData.AppEnum.CONTENT}`) &&
      !this._authService.isAuth
    ) {
      this._router.navigateByUrl(AppData.AppEnum.AUTH);
      return false;
    }

    if (
      state.url.startsWith(`/${AppData.AppEnum.AUTH}`) &&
      this._authService.isAuth
    ) {
      this._router.navigate([AppData.AppEnum.CONTENT]);
      return false;
    }
    return true;
  }
}
```

ДОДАТОК Г

Сервіси

Token service

```
export class TokenService {
  private readonly _key = 'token';

  private _userFields = [
    'email',
    'nickname',
    'password',
    'todos',
    'id',
    'role',
  ];
  private _token: string = '';

  constructor() {
    this.init();
  }

  public get(): string {
    return this._token;
  }

  public remove() {
    window.localStorage.removeItem(this._key);
    this._token = null;
  }

  public set(token) {
    this._token = token;
    window.localStorage.setItem(this._key, token);
  }

  /* PRIVATE HELPERS */

  private init() {
    const token = window.localStorage.getItem(this._key);

    if (this.validate(token)) {
      this.set(token);
    }
  }
```

```
  private validate(token: string): boolean {
    if (!token) {
      return false;
    }

    const tokenData = JSON.parse(token);

    let keyArr = Object.keys(tokenData);

    for (let field of this._userFields) {
      if (!keyArr.includes(field)) {
        return false;
      }
    }

    return true;
  }
}
```

Users service

```
@Injectable({
    providedIn: 'root'
})
export class UsersService {
    private readonly _baseApiRoute = 'http://localhost:5000/users';

    constructor(private readonly _http: HttpClient) { }

    public getAllUsers() {
        return this._http.get(this._baseApiRoute);
    }

    public putUser(user) {
      return this._http.put(`${this._baseApiRoute}`, user)
    }

    public getUserById(id: number) {
      return this._http.get(`${this._baseApiRoute}/${id}`);
    }

    public getUser(dto: Partial<AuthModels.User.IUser>):
Observable<AuthModels.User.IUser> {
        let params = new HttpParams();
```

```
        for (const key in dto) {
          params = params.append(key, dto[key]);
        }

        return this._http.get<AuthModels.User.IUser[]>(this._baseApiRoute, { params
}).pipe(
          map((users: AuthModels.User.IUser[]) => {
            if (users?.length) {
              return users[0];
            }

            throw new Error('404 => User not found!');
          })
        );
      }
}
```

Auth service

```
@Injectable({
  providedIn: 'root',
})
export class AuthService {
  private readonly _baseRegistrationApiRoute = 'http://localhost:5000/users';

  public get isAuth(): boolean {
    return !!this._tokenService.get();
  }

  constructor(
    private readonly http: HttpClient,
    private readonly _router: Router,
    private readonly _tokenService: TokenService,
    private readonly _usersService: UsersService
  ) {}

  public logIn(dto): Observable<unknown> { // dto type : AuthModels.User.IUser
    return this._usersService.getUser(dto).pipe(
      tap((user: AuthModels.User.IUser) => {
        this.onSuccessAuth(user);
      })
    );
  }
```

```
public logOut() {
  this._router.navigateByUrl(AppData.AppEnum.AUTH);
  this._tokenService.remove();
}

public register(formValue) {
  return this.getReg().pipe(
    map((users) => ({
      check: users.find((user) => user.email === formValue['email']),
      info: users,
    })),
    switchMap(({ check, info }) => {
      if (check) {
        return throwError(() => new Error('Provided email already exists!'));
      }

      return this.checkUsersExists({
        ...formValue,
        todos: [],
        id: ++info[info.length - 1].id,
      });
    })
  );
}

private onSuccessAuth(user: AuthModels.User.IUser) {
  const token = JSON.stringify(user);

  this._tokenService.set(token);

  this._router.navigateByUrl(AppData.AppEnum.CONTENT);
}

private checkUsersExists(dto: AuthModels.User.IUser) {
  return this.http
    .get<AuthModels.User.IUser[]>(this._baseRegistrationApiRoute)
    .pipe(
      map((item) => {
        item.length > 0
          ? (dto.role = AppData.Roles.USER)
          : (dto.role = AppData.Roles.ROOT_ADMIN);

        return dto;
      }),
      switchMap((registered) => this.createUser(registered)),
```

```
        tap((token) => {
          this.onSuccessAuth(JSON.parse(token));
        })
      );
  }

  private createUser(dto) {
    return this.http
      .post(this._baseRegistrationApiRoute, dto)
      .pipe(map((response) => JSON.stringify(response)));
  }

  private getReg() {
    return this.http.get<AuthModels.User.IUser[]>(
      this._baseRegistrationApiRoute
    );
  }
}
```

ДОДАТОК Д

Content Routing Module

```
const routes: Routes = [
    {
        path: '',
        component: ContentComponent,
        children: [
            { path: '', redirectTo: AppData.AppEnum.PROFILE, pathMatch: 'full'},
            {
                path: AppData.AppEnum.PROFILE,
                loadChildren: () =>
                    import('./modules/profile/profile.module').then(child =>
child.ProfileModule),
                    component: ProfileComponent
            },
            {
                path: AppData.AppEnum.TASKS,
                loadChildren: () =>
                    import('./modules/tasks/tasks.module').then(child =>
child.TasksModule),
                    component: TasksComponent
            },
            {
                path: `${AppData.AppEnum.TASKS}/:id`,
                loadChildren: () =>
                    import('./modules/tasks/tasks.module').then(child =>
child.TasksModule),
                    component: TaskPageComponent
            },
            {
                path: AppData.AppEnum.DESCIPLINES,
                loadChildren: () =>
                    import('./modules/desciplines/desciplines.module').then(child =>
child.DesciplinesModule),
                    component: DesciplinesComponent
            },
            {
                path: AppData.AppEnum.MESSAGES,
                loadChildren: () =>
                    import('./modules/messages/messages.module').then(child =>
child.MessagesModule),
                    component: MessagesComponent
```

```
            },
            {
                path: AppData.AppEnum.USERS,
                loadChildren: () =>
                    import('./modules/users/users.module').then(child =>
child.UsersModule),
                    component: UsersComponent
            },
            {
                path: AppData.AppEnum.GRADES,
                component: GradesComponent
            }
        ]
    },
]

@NgModule({
    imports: [RouterModule.forChild(routes)],
    exports: [RouterModule],
})
export class ContentRoutingModule { }
```

ДОДАТОК Е

Реєстрація ендпоінтів

```
const app = express();

const ports = process.env.PORT || 5000;

app.use(bodyParser.json());

app.use((req, res, next) => {
    res.setHeader('Access-Control-Allow-Origin', '*');
    res.setHeader('Access-Control-Allow-Methods', 'GET, POST, PUT, DELETE');
    res.setHeader('Access-Control-Allow-Headers', 'Content-Type, Authorization');
    next();
})

app.use('/users', usersRoutes);
app.use('/profile', profileRoutes);
app.use('/desciplines', desciplinesRoutes);
app.use('/tasks', tasksRoutes);
app.use('messages', messagesRoutes);

app.use(errorController.get404);
app.use(errorController.get500);

app.get('/api/graph', (req, res) => {
    const graphNodes = [
      { id: 'node1', label: 'Node 1', color: '#e18b8b' },
      { id: 'node2', label: 'Node 2', color: '#84e18b' },
      { id: 'node3', label: 'Node 3', color: '#8b9ce1' },
    ];

    const graphLinks = [
      { source: 'node1', target: 'node2' },
      { source: 'node2', target: 'node3' },
      { source: 'node3', target: 'node1' },
    ];

    res.json({ nodes: graphNodes, links: graphLinks });
  });

  app.post('/api/graph', (req, res) => {
    const { nodes, links } = req.body;
```

```
    res.json({ message: 'Дані графа збережено успішно.' });
  });

app.listen(ports, () => console.log(`Listening on port ${ports}`));
```