МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук

Кафедра Програмної інженерії

КВАЛІФІКАЦІЙНА РОБОТА

Пояснювальна записка

другий (магістерський)
(рівень вищої освіти)

Дослідження методів аналізу рекомендаційних систем для вирішення задач оптимізації використання часу

Виконав:

Студент 2 курсу, групи ІПЗм-20-1

Федорищев Д. В.

(прізвище, ініціали)

Спеціальність	121 – Інженерія програмного забезпечення
Тип програми	освітньо – наукова
Керівник	доц. Голян В.В.

Допускається до захисту

Зав. Кафедри _____ З.В. Дудар

Харків 2022

Харківський національний університет радіоелектроніки

Факультет	Комп'ютерних наук
Кафедра	Програмної інженерії
Рівень вищої освіти	другий (магістерський)
Спеціальність	121 – Інженерія програмного забезпечення
	(код і повна назва)
Тип програми	освітньо-наукова програма
Освітня програма	Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ:

Зав. кафедри _____

(підпис)

«___» _____ 202__ р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

студента Федорищева Дмитра Віталійовича

(прізвище, ім'я, по-батькові)

1. Тема роботи «Дослідження методів аналізу рекомендаційних систем для вирішення задач оптимізації використання часу».
затверджена наказом університету від «___» _____ 202__ р. №___
2. Термін подання студентом роботи до екзаменаційної комісії «__» ___ 202__ р.
3. Вихідні дані до роботи Рекомендаційна система, методи аналізу, поведінковий метод, метод колаборативної фільтрації, мова програмування Python, бібліотека TensorFlow2.
4. Перелік питань, що потрібно опрацювати в роботі мета роботи, аналіз предметної галузі та постановка задачі, аналоги, опис ідеї і алгоритму, методи аналізу, колаборативна фільтрація проектування і розробка програмного модуля, проектування і проведення експериментів, аналіз результатів.
5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, ілюстрацій (слайдів): мета роботи, постановка задачі, методи і алгоритми, опис отриманих результатів, інтерфейс програмної системи, демонстраційні матеріали.

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Пошук інформації та аналогічних рішень	25.09.2021	виконано
2	Розробка плану дослідження	18.10.2021	виконано
3	Вибір датасету для дослідження	1.11.2021	виконано
4	Вибір моделі для дослідження	6.11.2021	виконано
5	Аналіз предметної галузі та постановка задачі	20.11.2021	виконано
6	Опис ідеї та алгоритму	22.12.2021	виконано
7	Проектування програмного модуля	14.01.2022	виконано
8	Розробка програмної системи	25.01.2022	виконано
9	Проектування і проведення експериментів	22.02.2022	виконано
10	Аналіз отриманих результатів	25.02.2022	виконано
11	Підготовка пояснювальної записки	20.04.2022	виконано
12	Захист роботи	23.05.2022	

Дата видачі завдання _____ 202 р.

Студент _____
(підпис)

Керівник роботи _____ доц. Голян В. В.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка містить: 73 стор., 37 рис., 4 табл., 13 джерел.

ШТУЧНИЙ ІНТЕЛЕКТ, РЕКОМЕНДАЦІЙНА СИСТЕМА, АНАЛІЗ, ВЕБ-СИСТЕМА, TENSORFLOW, ІНСТРУМЕНТИ РОЗРОБКИ.

Ціллю цього дослідження є способи аналізу машинного навчання, а саме його алгоритми, підходи до вирішення задач, де за цільову групу беруть алгоритми, що мають давати рекомендації, виявлення слабких місць цих систем й їх модернізація шляхом створення нового сайту застосунку, що матиме відповідний програмний модуль.

За мету роботи взято аналіз алгоритмів та їх модернізація з подальшим створенням другої стабільної версії системи для підвищення професійних навичок. Це буде системи, веб-застосунок, що дозволить поглибити знання у сфері IT. Така система дасть відвідувачам сайту проходити курси на цьому сайті й буде давати такі рекомендації, що найкращим чином підійдуть для користувача, тому гість зможе економити свій час, замість того, щоб відвідувати навчальні засоби чи довго обирати навчальні програми й шукати курси.

Всі методи, що використовуються для розробки засновані на засобах навчання машин, алгоритмах та штучному інтелекті. Все написано на мовах: C#, Javascript, Python. Були використані фреймворки: ASP.NET Core, Tensorflow 2, Angular й була використана реляційна база даних MS SQL. Вся розробка проходила у середовищі Visual Studio, його версіях 2019 й Code.

За ціль роботи ставиться аналіз алгоритмів, їх модернізація й використання при створенні веб-системи й рекомендаційного модуля до нього, який рекомендує курси на основі алгоритмів машинного навчання. Така система дасть можливість значно ефективніше використовувати час та скоріше розвивати. Бо, насамперед, не потрібно буде лазити й шукати курси, також, можна зекономити час на шляху до університету чи навчального закладу, завдяки тому, що курси будуть пройдені на сайті.

ABSTRACT

Explanatory note to the undergraduate practice: 73 p., 37 fig., 4 table, 13 sources.

ARTIFICIAL INTELLIGENCE, RECOMMENDATION SYSTEM, ANALYSIS, WEB SYSTEM, TENSORFLOW, DEVELOPMENT TOOLS.

The aim of this study is to analyze machine learning, namely its algorithms, approaches to solving problems, where the target group is algorithms that should give recommendations, identify weaknesses in these systems and upgrade them by creating a new application site with a software module .

The purpose of the work is the analysis of algorithms and their modernization with the subsequent creation of a second stable version of the system to improve professional skills. It will be a system, a web application that will deepen knowledge in the field of IT. This system will allow site visitors to take courses on this site and will give the best recommendations for the user, so the guest will be able to save time instead of attending training facilities or choosing training programs and looking for courses.

All methods used for development are based on machine learning tools, algorithms and artificial intelligence. Everything is written in the following languages: C #, Javascript, Python. The following frameworks were used: ASP.NET Core, Tensorflow 2, Angular and the MS SQL relational database was used. All development took place in Visual Studio, its 2019 and Code versions.

The aim of the work is the analysis of algorithms, their modernization and use in the creation of a web system and a recommendation module to it, which recommends courses based on machine learning algorithms. Such a system will allow you to use time much more efficiently and develop faster. Because, first of all, you will not need to climb and look for courses, and you can save time on the way to university or school, thanks to the fact that the courses will be held on the site.

Я, Федорищев Дмитро Віталійович
(прізвище, ім’я, по-батькові)
студент групи ІПЗм-20-1 здобувач вищої освіти на другому (магістерському) рівні

кафедра програмної інженерії,
(повна назва кафедри)

заявляю: моя кваліфікаційна робота на тему
Дослідження методів аналізу рекомендаційних систем для вирішення задач оптимізації використання часу,

що буде представлена до ЕК для публічного захисту, виконана самостійно, в ній не містяться елементи плагіату і вона може бути опублікована в електронному архіві відкритого доступу ElArKhNURE. Всі запозичення з друкованих та електронних джерел мають відповідні посилання.

Я ознайомлений(а) з діючим положенням «Про протидію академічному плагіату в ХНУРЕ», згідно з яким виявлення плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту та застосування дисциплінарних заходів.

ЗМІСТ

ВСТУП

В усі часи навчання, знання, наука були і є найбільш важливою річчю в існуванні кожної окремої людини та сукупності їх. Завжди всі особи, протягом часу й вздовж свого життя безперервно щось опановують, що дає їм змогу покращити своє положення у соціумі, та збагатитись [1]. Наприклад, за більший стаж та кількість навиків професіонал отримуватиме, більшу платню. А надалі йому потрібно буде працювати менше часу, щоб забезпечити себе і це є прямою оптимізацією часу. За умови, що людина продовжує розвиватись, вона може не лише поглиблювати якусь одну навичку, а й отримувати досвід у чомусь новому, що позитивно вплине на її матеріальний стан й потенційні можливості подальшого розвитку. Ось як приклад, можна буде не лише вчитись машинному навчанню, а й оволодіти мобільною розробкою, а роль рекомендації полягатиме у тому, щоб надати курси з найбільш близькою мовою, щоб її було легше вивчити, та популярною, щоб на неї були вакансії, щоб були фреймворки й був певний комьюніті, що може допомогти у подальшому та не дасть цій мові занепасти.

Виникає запит, треба знайти учбові програми, курси, - це хороший спосіб відкрити більші можливості для себе, але потрібно витрачати все більше часу, оскільки на даний момент існує безліч різних областей знань в які можна поглиблюватись й які можуть бути потенційно цікаві чи приносити матеріальні блага. І постійно кількість таких галузей й напрямів зростає, що ще сильніше поскладнює можливість обрати щось одне чи взагалі зрозуміти, що було б цікавим й де це цікаве шукати [2].

Інтернет допомагає людям не лише сьорфити в інтернеті за кумедним контентом, придбати товари онлайн, шукати продукти та послуги, підтримувати соціальний контакт з ким-небудь, це й пряма можливість знайти наукові роботи, книги [3]. Всі пошукові, маркетингові й соціальні пошукові системи намагаються персоналізувати контент та рекомендації. Потрібно дослідити методи, які дозволять надавати рекомендації користувачам. Та дослідити методи оптимізації часу, завдяки цим самим рекомендаційним системам. Так, один з методів, що

використовується при наданні рекомендацій – це персоналізація. Загалом, цей метод, чи інакше кажучи підхід є доволі обширним. До його складу входять наступні методи:

- фільтраційний метод (конкретна, чи загальна);
- метод евристичного моделювання;
- поведінковий метод;
- метод колаборативної фільтрації.

Кожний з них має свої перевага й недоліки, котрі будуть частково розглянті у системі, але найбільш цікавим, популярним й спроможним до розкриття всієї потужності рекомендаційних систем й метод колаборативної фільтрації, що будуть детально розкриті у подальшому тексті роботи.

Побажання користувача є основним, що впливає на персоналізацію й рекомендації, що будуть на надані людині, яка використовує платформу з курсами. Щоб ця персоналізація працювала потрібно слідкувати за діями користувача у застосунку - кліків по курсам, лайків, часу, що людина вивчала щось. Портрет самої особи може бути створено, так система буде дивитись на тих, хто схожий за вподобаннями й на основі їх дій і інтересів буде пропонувати й новому юзеру за перших ознак подібности щось таке, що з найбільшою вірогідністю буде йому довподоби, а за метрику можна використати багато різних фіч чи їх комбінацій, починаючи з тих, що були описані вище.

Тому обладнання, методи, та тематика є важливими й актуальними нині й надалі. Створення програмного модулю, що надає рекомендації й створення самого програмного продукту є метою цієї атестаційної роботи, разом з тим, вивчення підходів до надання рекомендацій та методи їх аналізу є ще одним з напрямів цієї роботи. А такі рекомендації й застосунок для навчання онлайн дозволять значно оптимізувати використання власного часу за рахунок того, що не треба шукати якість навчальні програми у звичайному списку, а можна використовувати рекомендації. Й, до того ж, не треба їхати кудись. А можна всі ці курси проходити онлайн.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1 Аналіз предметної галузі

Звичайною практикою є використання всієї доступної інформації стосовно клієнта, наприклад, його особистих даних з профілю, приватних чи публічних джерел чи статистики його взаємодії з системою. За допомогою такого підходу не лише будуть згенеровані різні статистичні метрики, пов'язані з психотипом користувача, його інтересами, але й за допомогою цих даних можна пропонувати користувачу різноманітні освітні програми. Якщо вдасться створити метапрофіль клієнта, то це означатиме, що система повністю зрозуміє як зацікавите й як допомогти клієнту влучною порадою освітньої програми. Для корпорацій це означатиме збільшення доходу за рахунок рекламних інтеграцій та підписок на курс. А для користувача можливість отримати знання.

Можна персоналізацію й рекомендації категоризувати, можна залишити загальними, наприклад, ділення на стать, вік, регіон може надати деякої точності, що загалом гарно вплине на опит користування системою [4]. А за бали чи шкалу можна використовувати прямий підхід – це буде рейтинг, вподобайка, відгук, чи косвена – будь-яка взаємодія, наприклад, клік, перехід, наведення курсору.

Широкий спектр інструментів, технічних засобів й технік спеціалісти використовують, щоб розібратись з проблемою надання точних юзерських рекомендацій, загалом існують спеціальні алгоритмічні моделі й різні моделі машинного навчання. Варто виділити, що одним з таких підходів є колаборативна фільтрація, що є найбільш розповсюдженим з продвинутих, але звичайний пошук й фільтрація по категоріям також доволі часто використовується, іноді ці підходи комбіновані.

Якщо узагальнювати, то всі прості методи можна назвати «неперсоналізованими» і одним з варіантів таких систем, це мануальна статистична система, коли цілий відділ моніторить купівлю й розуміє, які продукти чи послуги найчастіше продаються парами чи групами. Далі ця статистика передається до IT-відділу і там хлопці роблять пряму логічну зв’язку між товарами і надалі вони

постійно з’являються разом у форматі «їх беруть разом». Такий підхід займає багато часу й може прогавити неочевидні комбінації через звичайний людський фактор. Тому це не є ефективним способом. Ще один спосіб – це звичайна фільтрація. Вона може бути різною, прикріпленою до «хітів продажу» чи звичайних категорій, або навіть до сезонних товарів. Якщо поглиблюватись до алгоритмів, то можна рахувати близькість й категорії за допомогою векторів, так обчислюючи кут у багатовимірному просторі можна також прорахувати близькість продуктів. Система, що використовує такий підхід є вже більш цікавою, але має доволі велику похибку, тому рекомендації, що будуть надаватись є далекими від істини й не є надійними. Отже, основну проблему не було вирішено.

А тут вже на допомогу приходить колаборативна фільтрація. Вона собою являє більш складну версію алгоритмічного підходу. Цей підхід можна зробити взявши за базу велику кількість різноманітних фіч. Можна сказати, що можуть бути застосована величезна кількість різноманітних стратегій реалізації. Якщо йти по таких стратегіях, то одна з них – це створення профілю користувача подібним до іншого користувача, так, якщо комусь щось сподобалось, то другій особі це також може підійти під вподобання. Такий підхід є дуже популярним, бо саме цей підхід надає можливість рекомендувати курси, товари, багато чого, взявши за основу дії і статистику інших користувачів. Як альтернативу можна розглядати інший, більш простий підхід, якщо його взяти до порівняння, то він, цей підхід, буде базуватись на подібності товарів чи послуг, на подібності курсів. І ця менша розумність показує, що підхід є менш цікавим й не буде у повній мірі покращувати рекомендації, у порівнянні, з іншими простими методами. Хоча зазначити, що така альтернатива є – варто. Так виходить, бо така реалізація алгоритму може бути порівнянна з простим підбором навчальних програм з тієї ж групи чи категорії, що навряд можна буде рахувати як інтелектуальні й влучні рекомендації, котрі користувач буде використовувати, а тому задача оптимізації часу не буде вирішена.

Щоб портрет користувача було створено можна користуватись часом, який цей користувач провів у курсі, а ще можна подивитись на те, які саме були їм вподобані. Якщо розгортати варіант використання часової фічі, то такою метрикою

влучно може бути проміжок скільки часу, який цей самий користувач знаходився на описі цієї навчальної програми. Тут за самий простий приклад можна взяти відкриття опису чи перегляду деталей навчальної програми, тобто, ми зможемо сміливо казати, що з вірогідністю 90% щось зацікавило користувача. Ось за один з прикладів, якщо назва курсу видалась йому цікавою, то з великою долею вірогідністю клієнт захоче дізнатись більше й піде читати опис, або пройти перший пробний відкритий урок. І тут можна безмежно розширювати джерела інформації, що будуть конвертовані у потенційні фічі для надання влучних рекомендацій.

Щоб далі використовувати ці фічі, варто використати інструмент, названий моделлю. Нині популярними є Darknet-19, MobileNet, Fast R-CNN, але цей напям стрімко розвивається, тому за тиждень від написання, рейтинг моделей може бути змінено. Тому, всі кто займаються науковою діяльністю у цьому напряму вивчають можливості покращення цих й інших моделей, пишуть наукові праці й звичайні статті, або просто виступають на конференціях з презентаціями. Машинне навчання є лідером популярності серед алгоритмів для розумного рекомендування, бо вони динамічно підлаштовуються під оточуючу середу. Але не варто забувати, що не лише алгоритми вирішують питання – датасети й їх обробка є половиною успіху. Так, щоб алгоритм машинного навчання мав якийсь успіх, потрібно мати великі обсяги точних даних, бажано без чи з мінімумом статистичного шуму. Так, обробка даних є ще однією складовою, а їх походження є неймовірно важливим. У цій роботі, буде використано датасет, створений вручну, бо неможливо отримати такі дані для новоствореної системи. Так всі науковці вкладають багато сил у створення як алгоритму, так й обробці та аналізу даних, але у випадку, кий розглядається у даній роботі, важливою слкдовою є правильне створення таких даних. І це є одним з лімітів у розроблюваній й аналізуємій системі, тому вірогідність похибки, через відсутність даних може бути значною, порівняно з ситуацією, коли є правильний датасет.

Алгоритмічні підходи, які були створені не на основі і не з використанням нейронних мережей, ось такі як ті, що зроблені на основі статистичних підходів, типу, як, комівояжер й його аналоги, також широко використовуються для того,

щоб розібратись з проблемою надання рекомендацій. Та попри те, що вони використовуються, вони за популярністю поступаються машинному навчанню, так знов, мою зазначити, що всі такі алгоритми, чим точніше вони будуть працювати, тим більш місткими вони є, а саме тому потребують все більших потужностей, яких у домашніх умовах неможливо досягти. І це є ще одним обмеженням системи, створювана нейронна мережа буде підлягати такому обмеженню, що означає, створення більш простої її версії, що також вплине на точність надання рекомендацій.

Коли підсумовуємо весь проведений аналітичний огляд предметної галузі, то бачимо, що лише деякі інструменти дають бажану точність й дають предоставляють потрібну ефективність, хоч мінімальну. Далі, бачимо, що якщо використовуються алгоритми машинного навчання, то результат зазвичай є більш точнішим, але такі рішення підлягають значним потребам, яй технічними, так й інформаційними. А у випадку некомерційної роботи, вони суттєво обмежують вихідну потужність й точність цієї роботи.

Зробімо поглиблення у проблему та аналіз аналогів.

1.2 Поглиблення у проблему та аналіз аналогів

Перш ніж починати розглядати вже давно існуючі аналоги, варто сказати, що існує безліч систем, які можуть рекомендувати послуги, продукти, курси й кількість таких систем лише зростає. Зазвичай, всі такі системи обмежені у галузі застосування й дуже сильно переплетені з конкретним доменом й доменою логікою. Таких систем й дуже багато, наступні наведені нижче – це найбільші гравці на ринку, чи перші за пошуковими запитами у нашому регіоні, й повністю відповідають домену підвищення кваліфікації. Щоб оцінити роботу моделі використовувалась тестова частина датасету й метод сліпої оцінки, коли тестувальник взаємодіє з системою через юзерський інтерфейс, й не має доступу до коду застосунку. Нагадаю, що це обмеження було умисно зроблено саме для тестування, бо багато систем саме так й тестуються. Протягом ходу поглиблення в аналоги, було оброблено безліч інформації з мережі інтернет, проведено аудит

багатьох для підвищення професійних навичок. Відбувся процес аналізу аналогів, переглянуті їх позитивні (сильні) й негативні (слабкі) сторони, сторони цих застосунків, з акцентом на рекомендаційний модуль.

Всі успішні аналоги, що були оброблені є приватними, тому поглянути на них «з середини» не є можливим. Вся оцінка проводилась за допомогою відкритих джерел й з досвіду використання. А з цього постає висновок, що не є можливим зрозуміти що за підходи чи машинне навчання застосовувалось під час розробки й використовується надалі для того, щоб рекомендувати освітні програми, курси. Саме ці системи будуть розглянуті у наступних розділах, а потім буде проведено їх порівняння.

Оціненими будуть точність рекомендацій (їх відповідність до створенного профіля користувача й їх влучність до переглянутих курсів), будуть оброблені публічні джерела з доступною інформацією щодо підходу рекомендування й алгоритмів, також частина часу буде приділена читанню й обробці документації цих систем, з метою виявлення принципових нюансів у підходах.

Перейдем до першого аналогу.

1.2.1 Coursehunter

Це є сервіс, що розрахований для фронтенд та бекенд розробникові, здивувало, що на момент написання він не містив курсів з мобільної розробки, адже це також один з напрямків для самовдосконалення й надає можливість заробити гроші [5]. А сама мобільна розробка є популярним напрямом, який багато де переважає й є більш популярним за звичайні веб-сайти. Але, у них, мабуть, така політика, чи були зламані фільтри, таке теж буває.

Це один з таких ресурсів, де іноді можна знайти безкоштовні освітні програми, але більш важливою частиною є платні, преміум, підписки, що надають доступ до усіх курсів, та відкриває можливість дивитись весь контент та у кращій якості. Можливість прямого контакту з преподавателями також є невід’ємною перевагою.

Коли юзер відкриє головну сторінку, то його зустріне наступний інтерфейс, що наведено на рисунку 1.1.

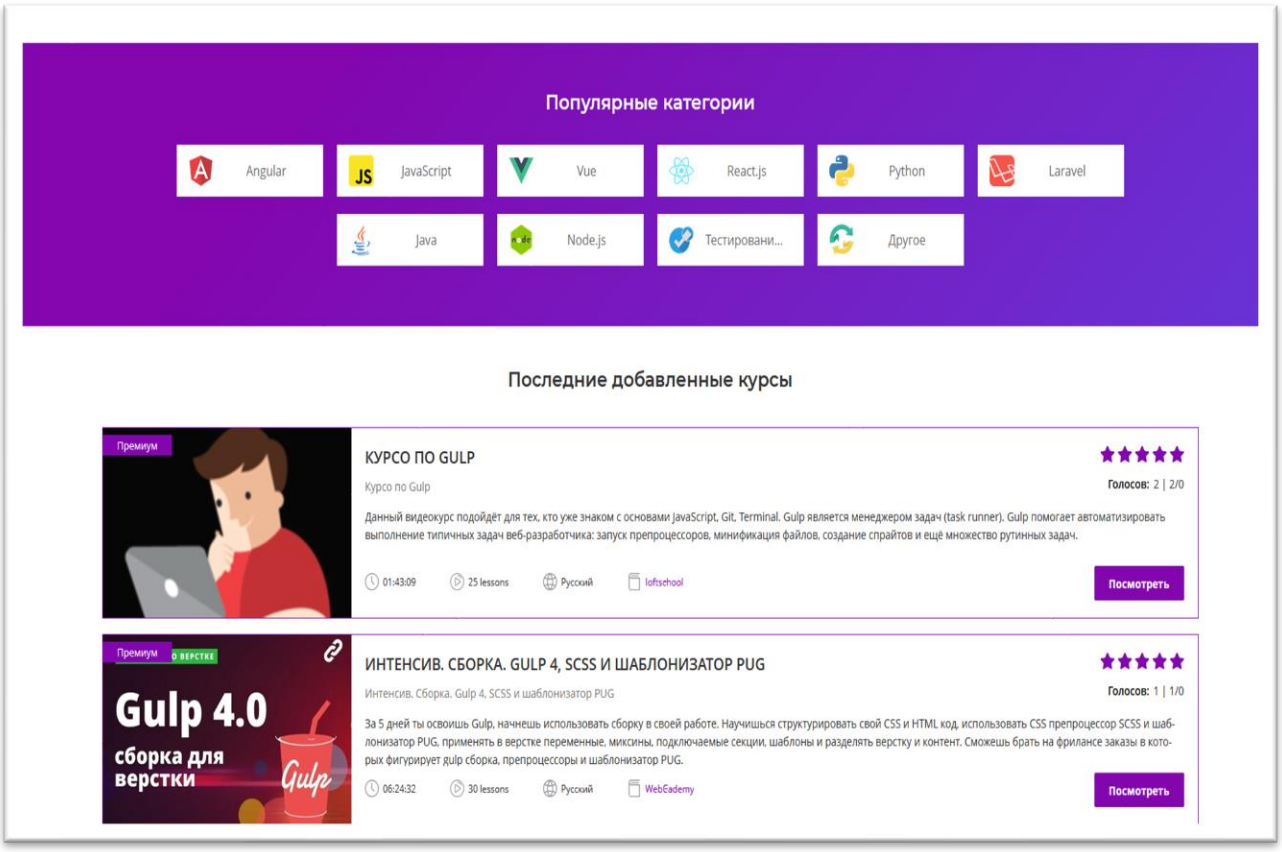

Рисунок 1.1 – Перша лендінгова сторінка сайту Coursehunter

Переходячи до розгляду й аналізу модуля надання рекомендацій – він є банальним, вся суть заключається у тому, щоб відобразити базові категорії на першому екрані. Всі ці групи є статичними, та виставляються мануально на веб-сайті, тому модуль рекомендацій тут можна покращувати й розвивати, а на нього посилатись як на гарний приклад – неможливо, лише як на аналог, що потребує допрацювання.

Перейдемо до другого аналогу.

1.2.2 Coursera

Stanford заснував й веде, руками багатьох викладачів, таку платформу, як Coursera [6]. Вона є онлайн додатком, що програми навчання, щоб люди могли

проходити висококласні курсі з будь-якого місяця на планеті й не були фізично прив’язані до учбового закладу.

Пердша пейджа сайту є наведеною на рисунку 1.2.

Рисунок 1.2 – Лендінг Курсери

Головною особливістю цього ресурсу є те, що всі викладачі мають бути сертифікованими, а більшість з них напряму пов’язані з найпопулярнішими учбовими закладами, а це означає, що сертифікати визнаються всюди й він є першим у цьому серед усіх аналогів. Тут також можна навчатись безкоштовно, але найкращі курсі є платними, та іноді бувають акції, що дадуть можливість отримати щось платне майже за безцінок, чи взагалі безкоштовно. Ця платформа містить текст, слайди, відео, онлай-чати. Лише Coursera надає програму MasterTrack, що дозволяє вивчати освітні програми світових вищів з дому, а це вже є деякою економією власного часу на дорогу.

Переходячи до найбільш цікавою, наразі, частини – рекомендацій. Coursera не має такого модуля, він її не потрібен, бо зазвичай її відвідувачі направлені на конкретні освітні програми, але це водночас є ще й потенціалом для розвитку й масштабування. Тому, хоч якими б плюсами ця платформа не обладала, вона не може одностайно отримати первінство у своєму домені.

Перейдемо до третього аналогу.

1.2.3 Udemy

Онлайн-конструктор курсів, що є одним з лідерів за популярністю слугує Udemy [7]. Ця платформа дає змогу викладачам будувати курси прямо онлайн, наживо, лише завантажуючи свої матеріали на відповідні сторінки. Так можна легко завантажити різні формати даних, а Udemy допоможе розпарсити, чи вставити такі файли напряму у потрібні місця. Так, електронні книги, PDF, аудіо матеріали, архіви, презентації та багато іншого може бути завантажено й взяту за основу при створенні курсів. Сам сайт дозволяє також прямий контакт з викладачем та працювати з доскою онлайн, прямо під час заняття.

Сам спектр освітніх програм є необмеженим, бо кожний бажаючий може навчати чомусь своєму. Ось, професор медичного вищу може завантажити свої лекції й тепер читач зможе проходити лекції з медицини, гістології, біології, тощо. А водночас з ним, з викладач іншого вищу вже буде завантажувати лекції з програмування. І цей список можна продовжувати безкінечно.

Ця платформа є першою за популярністю, вона має доволі добрий рекомендаційний модуль, який одночасно охоплює різні напрями від машинобудівництва до готування, та може мімікрувати рекомендації залежно від вподобань юзера доволі добре. Посилаючись на гугл аналітику та публічні джерела, можна казати, що понад 50 мільйонів юзерів має ця система, вона трохи поступається Курсері у визнаності своїх сертифікатів, але все ж не сильно, тому дипломи тут цінуються.

Цікавою й визначною частиною функціоналу є створення власних ступеней розвитку, де проходячи різні курси можна оволодіти доволі широким спектром знань. Велика кількість великих IT-компанії зацікавлені у такому підході та самі власноруч пропонують подібне своїм працівникам, мотивуючи їх грошима, бонусами, а тут таких підход йде за замовчуванням, що дуже добре.

Опросник, який гість заповнює дає базове розуміння, у якому напрямі давати рекомендації, що можна взяти за приклад й додати в одній з ітерацій до цього додатку.

Першу сторінку цього сайту наведено на рисунку 1.3.

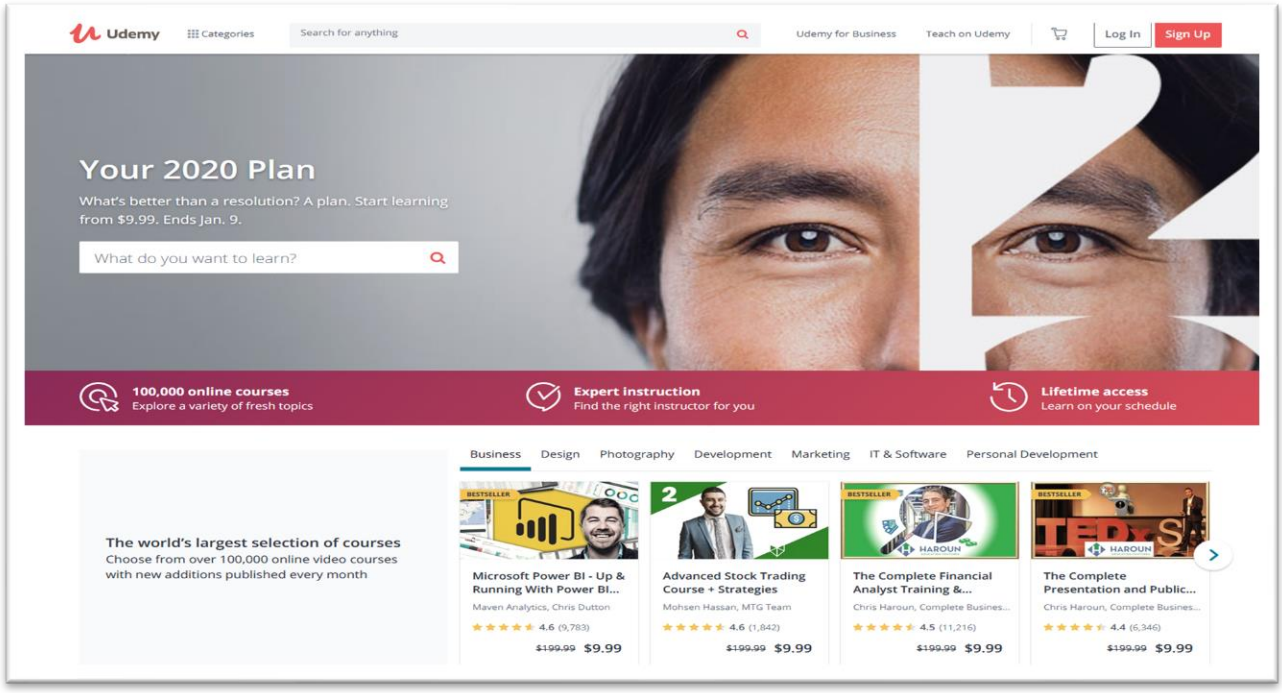

Рисунок 1.3 – Перша сторінка сайту Udemy

Переходячи до мінусів, те, що сайт вимагає заповнити такий опросник має й негативний посил, бо це зайвий час, коли гість хоче переглянути щось конкретне, а за основу рекомендацій можна брати щось популярне й модифікувати перший час вподобання з великою дельтою, що дасть більше точності й покращений юзер експіріенс.

Перейдемо до четвертого аналогу.

1.2.4 Metanit

Дивлячись на найпростіші аналоги – це Metanit [9]. Він є дуже простим сайтом, що містить безкоштовні копії документації з розширеними прикладами, чи іноді навіть деякі вузькоспеціалізовані курси. Вся адаптація зроблена російською, бо це найбільш розповсюджена мова у нашому сегменті інтернету, а, як було

згадано раніше, це також є важливою фічею й має бути згадано й використано, при аналізі й можливому наданню рекомендацій.

Далі, якщо подивитись на рисунок 1.5, то можна побачити першу сторінку веб-сайту.

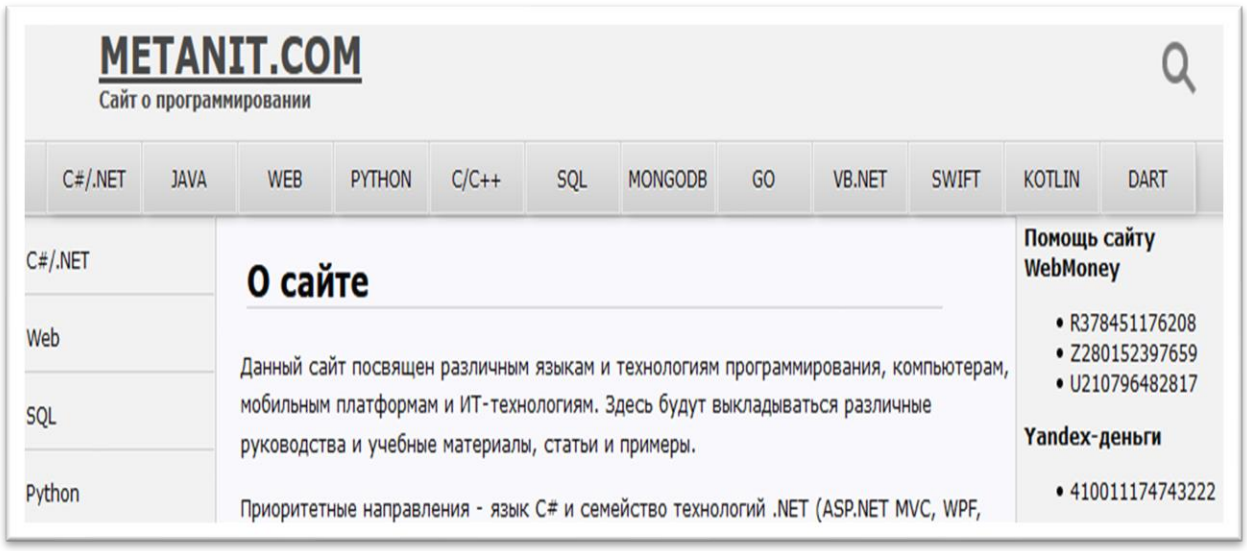

Рисунок 1.5 – Перша сторінка веб-сайту Metanit

Переходячи до найцікавішого, до рекомендацій, можна одностайно сказати, що тут немає нічного, окрім меню з конкретними розділами за мовами, іноді, зустрічаються технології. Тому, він має вузьку спеціалізацію, але, все ж, є наразі популярним, тому його теж варто оцінити й мати на увазі, як потенційного конкурента.

Перейдемо до останнього аналогу.

1.2.5 Stepik

Ще однією платформою є Stepik [8], це онлайн веб-сайт, на якому також можна створювати курси для неліцензованих викладачів, за більш простої, але менш зручної схеми взаємодії, аніж у Udemi, та можна вивчати матеріал онлайн на цій платформі. Ця платформа використовується все ширше, ніж аналоги, але формат використання є доволі специфічним й має вузький профіль – це онлайн змагання з алгоритмічного програмування. Її іноді використовують для офлайн

змагань, але зазвичай, це все ж таки онлайн платформа, яку всі використовують віддалено просто для навчання.

Інтерфейс Степіку можна побачити на рисунку 1.4.

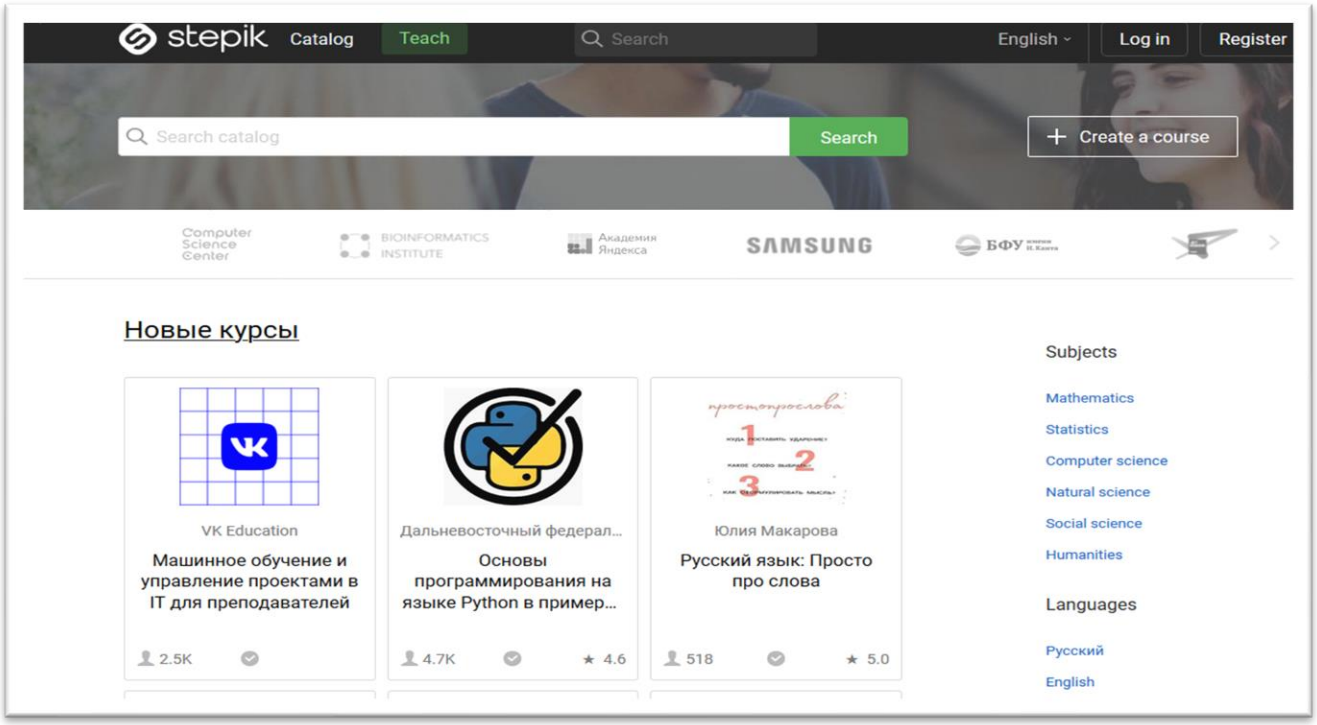

Рисунок 1.4 – Головна сторінка сайту Stepik

Досить зручним є можливість використання чату, але він є у всіх крупних гравців на цьому ринку, на цьому домені, тому не є чимось особливим, лише звичайна даність. Степік несе у собі той же самий принцип індивідуального персонального підходу до рекомендацій й учбового плану, що надається кожній людині, це означає, що рекомендації тут мають деяку інтерактивність й трохи підлаштовуються під кожну людину, але це відубвається доволі просто, через відвідування категорій.

До того ж, переглядаючи головну сторінку помітно, що всі рекомендації надаються чи за категоріями, як було загадано вище, чи загалом інформація береться зі статистики й на її основі першим у статичній стрічці зображається курс, що є першим за переглядами, тобто модератор сайту власноруч, чи за допомогою

мінімальної автоматизації оновлює ці рекомендації, що є доволі затратним процесом й не несе інтерактивності.

Перейдмо до порівняння.

1.2.6 Фінальне порівняння наведених вище систем

Ну і тепер час переходити до підбивання результатів аналізу цих рекомендаційних систем, на яких студент, чи, взагалі, будь-хто може навчатись. Зараз, підсумовуючи всю проділану роботу, варто зазначити, що наступні аналоги були розглянуті: Coursehunter, Coursera, Udemy, Stepik, Metanit, котрі мають свої позитивні (сильні) й негативні (слабкі) сторони. Чи, інакше кажучи, переваги й недоліки.

Далі всі дані будуть впиані для порівняння у табличний формат й фінальне підсумування результатів можна побачити у таблиці 1.1.

Таблиця 1.1 – Порівняння характеристик аналогів

Додаток	Надає рекомендації	Профіль є інтелектуальним	Ступінь визнанності дипломів	Комфорт платформи
Coursehunter	+	-	Лише на російському ринку, мала	Мала
Coursera	+	-	Максимальна	Максимальна
Udemy	+	+	Середня	Максимальна
Stepik	+	+-	Середня	Середня
Metanit	-	-	-	Мала

Ще раз варто зазначити, що, й буде зназначено, що аналіз проводився у форматі чорної скриньки, з деяким підгляданням на доступні матеріали з теми, бо більшість інформації, особливо, що напряму поєднана з рекомендаціями є інтелектуальною собственістю.

Тепер, переходимо до аналізу таблиці 1.1. З неї виходить, що власне рекомендації надають не всі, але хоч у деякому вигляді їх надає більшість. Далі, поглиблюючись у те, які це сами рекомендації – вони можуть бути статичними, тобто не інтелектуальними. Якщо зважувати їх інтелектуальність, то є явний лідер. Далі, дивимось на визнаність дипломів, загалом вона «деяка», але є система, де взагалі немає сертифікатів у будь-якому вигляді, а є така, де визнаність, буквально, всесвітня. Не менш важливим є комфорт використання платформи – тут все доволі однозначно, чим це більший гравець на ринку, тим більш продуманий їх додаток, що пояснюється бюджетом та статистикою.

Тепер варто оформити цілі.

1.3 Оформлення цілі

Що ж, починаючи з першого, необхідного етапу изначення потреб до системи, тобто зафіксємою базові вимоги, котрим ця система має задовільняти, для того, щоб було розуміння над чим потрібно працювати. Якщо розмислювати, то вимогами можна назвати сукупність характеристик та валстивостей додатку, якими необхідно наділити модуль чи їх сукупчення. Отже, трохи згодом, вони й будуть наведені у перечні, й ці властивості системи будуть наявні й притаманні системі.

А тепер, кількі слів про саме мету цієї роботи. За ставиться аналіз предметної галузі з ціллю виявлення методів аналізу рекомендаційних систем, де за приклад беруться системи підвищення професійних навичок, тобто, IT-сферу курсів, щоб зрозуміти, як такі системи, такі рекомендаційні модулю можуть покращити життя звичайного користувача та допомогти йому зекономити час на підбір учбових програм, й, також, важливим, але вторинних, є економія часу на дорогу до учбових закладів. Додаючи осязання цій меті, буде створено програмний модуль, що надає рекомендації, а також веб-сайт, на якому цей модуль можна буде тестувати, та побачити у потенційній середі використання.

Далі, переходячи до очікуваної поведінки веб-застосунку, очікуються наступні ролі:

- гість – людина, що вперше зайшла до сайту, має доступ лише до авторизації, реєстрації та перегляду головної сторінки, щоб лише базово ознайомитись з контентом;
- юзер – людина, що вже має профіль, та може його заповняти, відвідувати курси, ознайомлюватись з їх деталями, отримуватиме рекомендації;
- адмін – може редагувати старі й додавати нові освітні програми, також такий користувач може створювати звіти, на основі поведінки клієнтів.

Тому, вище було наведено те, що було згадано раніше, о потребах у додатку. Вони були сформовані та розписані по ролях. Ці ролі мають бути реалізовані як на фронтенді, та й на бекенді. А модуль рекомендацій має бути тісно пов’язаним, але фізично окремим від бекенду.

2 ОПИС ІДЕЇ ТА АЛГОРИТМУ

2.1 Опис проблеми яку вирішує програмний продукт

Дослідження виявили серйозні витрати часу та ресурсів, коли людна вимушена шукати інформацію на новому, чи, навіть, знайомому сайті. Така втрата часу в середньому становить 1,8 години щотижня, а якщо людина не впевнена що їх цікаво, то й всі 4.7 години лише для того, щоб знайти цікавий курс для вивчення. Це від 5% до 10% від вільного часу середньостатистичної людини на тиждень. Коли курси не можна легко знайти або організувати таким чином, щоб клієнти могли легко знайти їх та продивитись аналоги, щоб обрати щось це збиває настрій та заважає концентрації на навчанні. І це лише для проходження якогось одного курсу.

Частково вирішують проблему пошукові системи, але вони слабо враховують індивідуальні потреби користувача [10]. Тому для вирішення цієї проблеми були створені рекомендаційні системи, які фільтрують та сортують для користувача інформацію. Вони використовують думку спільноти, щоб допомогти у виборі цікавого контенту з величезної кількості варіантів. Коло завдань, які вирішуються рекомендаційними системами, постійно розширюється. Сучасні рекомендаційні системи покращують точність своїх оцінок, враховуючи контекст та семантику.

Варто зазначити деякі відомості.

2.2 Відомості стосовно обраного підходу

Що ж, переходячи до реалізації, варто сказати, що її було почта з будування рекомендаційного модуля, котрий являє собою нейронну мережу, деталі про які будуть наведені у подальшому. Також, перед початком власне створення нейронної мережі, було проведено аналіз підходів й виявлено правильні алгоритми й ідеї для створення цієї самої нейронної мережі. Вхідними даними для отримання рекомендації є час, який людина провела на курсах, сумарно на всіх лекціях й описах курсу, масштабовані до розмірностей від 0 до 1. Така інформація є потрібною, бо варто від чогось йти, коли створюється прогноз, у кейсі роботи –

рекомендація. Ще одним параметром є проставлені вподобайки на курси й конкретні уроки, це дозволить ще розширити профіль користувача, з ціллю покращення рекомендацій. Для того, щоб навчити нейронну мережу думати, варто десь взяти дані, ці дані є замоканими і напряму витягуються з бази даних системи, поступово таке рішення можна буде масштабувати чи модернізувати, але в даному випадку цього достатньо, за умови відповідності всім лімітам системи.

Тепер поглянемо на рисунок 2.1, на якому зображено орієнтовну схему, за якою працює й вчиться нейронна мережа, зауважу, що схема узагальнена.

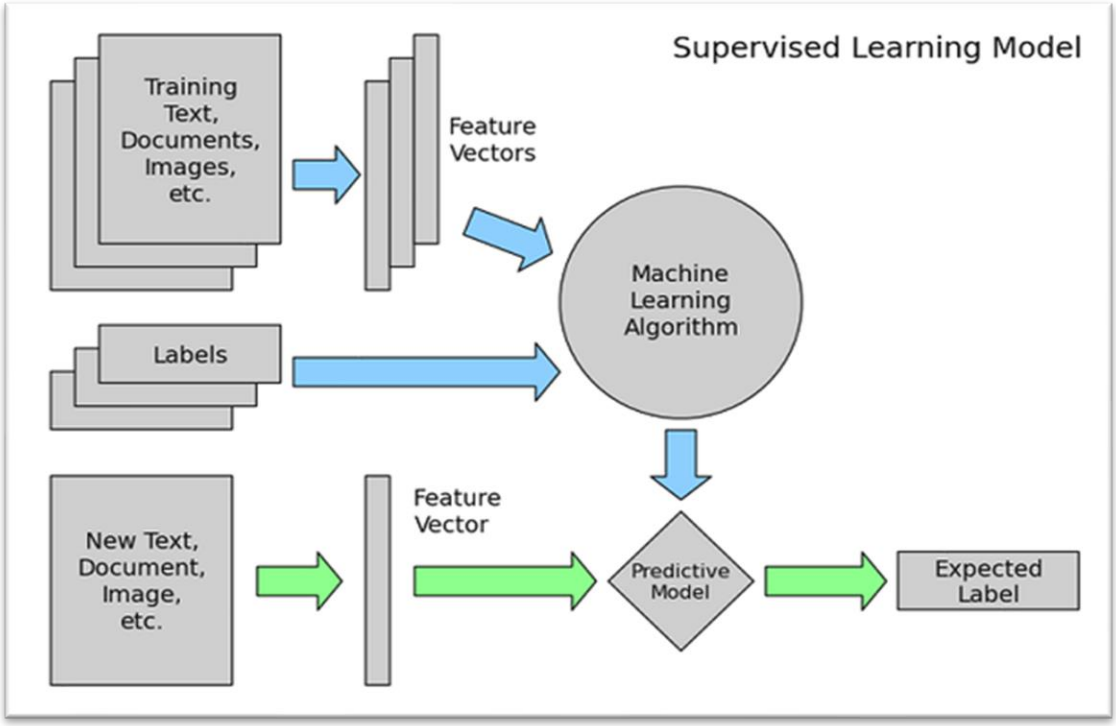

Рисунок 2.1 – Узагальнена схема взаємодії з нейронною мережею

Ітеративно, коли накопичується деяка статистика на сайті, тобто люди проходять й відвідують більше курсів, ставлять свої лайки, вподобайки, на курси й конкретні уроки, з’являється більше інформації, за якою можна будувати нейронну мережу, під будуванням мається на увазі в тому числі й перенавчання, оновлення нейронних зв’язків іншими вагами, що дасть точніші рекомендації. Трохи поглиблюючись у технічну частину, важливу роль відіграє модуль Vectori, який трансформує вхідні дані до відповідного вектору, а цей вектор використовується як

під час навчання, так й при наданні саме рекомендацій. Як було зазначено раніше, час та вподобайки конвертуються до відповідних векторів й вони далі використовуються у тому форматі, який був зазначений вище.

Що є приємним у роботі з нейронною мережею – це те, що можна різноманітним чином модернізувати її, змінювати деякі сетинги, це дозволить протестувати гіпотези щодо збільшення точності надання прогнозу. Одним з шумів, котрі є деяким лімітом системи, є наступна проблема, невідомо, чи юзер справді був зацікавлений у курсі, чи він забува закрити сторінку коли раптово клікнув.

Але це є очікуваним обмеженням і зазвичай користувачі не звикли надовго лишати вкладки, тому цей варіант і був обраний, як ключова метрика й емпіричні досліди довели, що це не дарма. Подитожу, що загалом було доведено, що ця статистична похибка має вплив, але занадто малий, щоб з нею серйозно рахуватись. Та, чим більше даних статистичних буде у системі, тим краще такі шуми можуть бути виявлені й придушені. Але це буде вже згодом.

Ще варто зауважити, чому такий підхід був обраний, він був обраний тому шо у багатьох соревнуваннях такі підходи отримували призові місця, й про це було багато інформації, що цій роботі дуже допомогло у подальшому розвитку цієї теми. Але однією зі складностей було порівняння даних з іншими системами й тестування алгоритму, адже комерційна таємниця крупних проектов забороняє їм надавати подробиці роботи власних систем.

Варто зазначити про дані.

2.3 Дані, їх презентація та обробка

Тепер слід перейти до методів, що були використані. Один з таких методів – це наївний баєсів [11] класифікатор, що являє по своїй суті такий класифікатор, що працює з вірогідностями. Цей класифікатор за основу бере, як зрозуміло з назви – співзвучну теорему, а по своїй суті він вимірює наскільки один елемент належить до групи спостережень, тобто, іншими словами, до іншої групи. Так, це дозволяє зрозуміти деяку згрупованість та подібність, що далі можна використовувати для

того, щоб зрозуміти наскількі курси є подібними, а цю інформацію можна покласти до основи тестування нейронної мережі.

Суть цього алгоритму, саме у тому, що він скаже процент подібності між елементом та кластерами, а не лише розмістить його у ньому. В разі рівнозначної подібності можна буде отримати такі значення, що дадуть це зрозуміти, тобто, відсоток подібності буде однаковим. Одним з недоліків є те, що шуми будуть впливати на вимірювання, бо за основу береться ідея, що все оточення константе та не впливає ніяк на систему. Але переваги все ж переважають – значне прискорення роботи алгоритму, до експонента перетворюється на звичайну пряму пропорційність до розміру вектору.

Коли розмова йде до вимірювання точності прогнозу, то тут не можна бути одностайним. Немає метрик, що дозволять при існуючих лімітах точно виміряти, то і цій роботі вони будуть проходити емпіричну оцінку на задовільність наданих рекомендацій, дивлячись на інтереси створеного тестовог профіля та тестові дані. Але це вже було зазначено у розділах вище, в тих кількох системних обмеженнях. Насамперед, тут вспливає обмеження з даними, датасетом, якого немає і який може виникнути лише після деякого значного часу користування системою. Тому, саме емпіричний підхід, що базується на моїй експертності й буде використано.

Тепер варто перейти до алгоритму роботи.

2.4 Теоретичні відомості про алгоритм роботи

Тепер вже варто перейти саме до виведених кінцевих формул, що є теоретичною основою роботи алгоритму, котрий повністю реалізовано у програмному модулі бібліотеки TensorFlow 2. Так саме вони, саме ці формули, й є наведені далі у роботі. А якщо буде цікаво поглибитись у деталі інструментарію, що використовується, то всі зазначені назви алгоритмів та методів, можна буде дуже легко знайти за посиланнями, тобто у літературі, на яку є посилання, чи якщо банально прогуглити їх у мережі інтернет.

Так, перший метод – це метод на основі подібності елементів. Для початку, для кожного елемента j обчислюється міра близькості до елемента i. Для цього можна використовувати, наприклад, коефіцієнт Пірсона:

$$s_{i,j} = \frac{\sum_{u \in U}(r_{u,i} - \bar{r}_i)(r_{u,j} - \bar{r}_j)}{\sqrt{\sum_{u \in U}(r_{u,i} - \bar{r}_i)^2}\sqrt{\sum_{u \in U}(r_{u,j} - \bar{r}_j)^2}}$$

де $s_{i,j}$ – це близькість між оцінкою i та j,

де $u \in U_i \cup U_j$ – безліч користувачів, які оцінили елементи i та j,

де r_i – це і-та оцінка користувача, тобто фіча.

Ось, наприклад, як деяка вподобайка чи часть перегляду курсу, котрий вже було масштабовано до єдиного формату величин.

Далі, вибираємо безліч елементів S, найближчих до об'єкта i. Судячи з експериментів, достатні результати виходять при k = 30 елементів множини S.

Слідом, варто пройти етап прогнозування рейтингу (оцінки) об'єкта на основі рейтингів близьких до нього об'єктів:

$$\bar{r}_{i,j} = \frac{\sum_{j \in S}(s_{i,j} r_{u,j})}{\sum_{j \in S}|s_{i,j}|}$$

Де всі величини є такими ж, як було зазначено під час розбору формули трохи вище.

Цей алгоритм відбиває теоретичну основу методу, але практично ряд чинників вимагає переосмислення розрахунків. Як правило, переважна більшість оцінок невідомо, і розрідженість матриці оцінок досить висока. З іншого боку, дані, які вже є в матриці досить суб'єктивні. Деякі користувачі – оптимісти, та його оцінки завжди високі (середнє 4 із 5), інші користувачі – циніки, їх оцінки завжди занижені (середнє 2,5 із 5). Крім цього, завжди є елементи, що подобаються всім. З метою боротьби з припасуванням розріджених даних з оцінками, проводиться регулювання моделей таким чином, щоб скоротити ймовірність появи випадкових зв'язків між оцінками, які не відображають дійсність. Регулювання контролюється

константами, що позначаються як $L_1, L_2, \ldots$. Точні значення цих констант визначаються перехресною перевіркою. У міру їхнього зростання, регуляризація стає дедалі важчим. Для того, щоб оптимізувати продуктивність видачі рекомендацій, важливо нормалізувати оцінки до обчислення матриці подоби [12].

Це може бути досягнуто шляхом обчислення базового прогнозу, в якому інкапсулюють відхилення користувача та елемент. Пари користувач-елемент (u,i) для яких оцінки $r_{u,i}$ відомі складають безліч K. Базовий прогноз для невідомої оцінки $r_{u,i}$позначається $b_{u,i}$та визначається формулою:

$$b_{u,i} = O + b_u + b_i$$

де O - загальна середня оцінка,

де b_u і b_i - параметри, які показують відхилення користувача u і елемента i відповідно від середнього значення.

3 ОБРОБКА МЕТОДІВ Й ПІДБІР ЗАСОБІВ

3.1 Огляд

Переходячи до огляду того, що буде в деталях покривати у наступних оглядах. Невеликий тизер – це буде саме те, що написано у тайтлі, тобто, будуть аналізуватись методи й вони будуть порівнюватись, будуть обрані засоби для роботи, паралельно ставлячи план на подальшу роботу. Так, протягм первинного етапу роботи з постановки требувань, було виявлено деяку закономірність, її сенс заключається у великих часовитратах кожного користувача, на пошук курсів.

Так, користувач, мав витрати декілька годин, щоб лише зрозуміти, що йому було б цікаво вчити, але коли перейшов до курсу, міг почати його вивчати, навіть придбати, а він виявлявся не зовсім підходящим, а це вже була пряма трата обох – часу й грошей з його боку. Це неприпустимо. Тому, дуже зручно було б дати можливість користувачу чи повернути гроші за курс, якщо він йому не сподобався, а для цього можна використати відмітку про проходження уроку, й, якщо хоч якась значна частина курсу пройдена, то вважати, що все подобається і це є гарним штрихом, який імплементовано у додатку.

Далі, варто зазначити якого вигляду матиме система всередині, вона матиме: фронтенд додаток через який користувач буде взаємодіяти з додатком, бекенд додаток який відповідальний за обробку даних та предоставління певного роду проксі, перед взаємодією з модулем рекомендацій, саме рекомендаційний модуль, який дає рекомендації та останній, але не менш важливий елемент – база даних, котра на перших етапах розвитку системи може перебувати на одному кластері, чи навіть на одному сервері з бекендом. Тобто, коли клієнт відкриває сайт, то він бачить головну сторінку, яка дає йому рекомендації, що базуються на трендах, бо юзер є гостем.

Далі, після створення профілю, цей юзер бачитиме блок рекомендацій, що буде динамічно оновлюватись під його потреби, а його потреби визначаються з найбільшою чутливістю першим часом. Це зроблено для того, щоб якнайшвидше

зрозуміти як зробити юзер експіріенс цієї людини максимальним, задля того, щоб цей користувач був щасливий такою оптимізацією часу.

Така оптимізація часу й є кінцевою метою проводимого аналізу й створюємого на її основі додатку, що превзойде аналоги, особливо з позиції рекомендацій, особливо у випадку запуску платформи під комерційними прапорами та при збільшенні команди до хоча б кількох сотень осіб.

Тепер варто зобразити структуру за допомогою UML.

3.2 UML планування

Всі знають що таке UML, але ще раз – це інструмент, іноді кажуть мова, за допомогою котрого можна творити діаграми, які описують структуру системи, взаємодії елементів, ролі користувачів. Загалом вони нам дають повне описання й характеристику всього, що буде відбуватись ззовні та й усередині в розроблюваній системі.

Окремо варто відмітити, що дуже зручним є те, що опис буде візуалізованим, бо графіки й діаграми рисуються, й рисуються у комп'ютері, для наглядності, тобто їх так дуже зручно сприймати, а за використанням деяких онлайн систем, ще й редагувати. Так, візуалізація дасть можливість оцінити що відбувається у системі у кілька разів швидше, ніж перегляд сухих текстових даних.

Варто ще зазначити о рівнях абстракції, котрі UML дозволяє мати й створювати – вони стандартовані, що є дуже зручним у роботі. Було створено такі діаграми, щоб пояснити взаємодію елементів та загальну роботу системи:

- варіантів використання;
- класів;
- розгортання.

Кожна з цих діаграм пояснює окрему ідею, що був у всій системі, окрему її частину. Але вони загалом формують повний список всього, що відбувається у системі. Звернімо увагу на рисунок 3.1, там було зображено першу з цих діаграм, вона називається «варіантів використання», й, як можна судити з назви, на ній

описані варианти використання, у форматі, який тип юзеру й до чого, до яких дій має доступ.

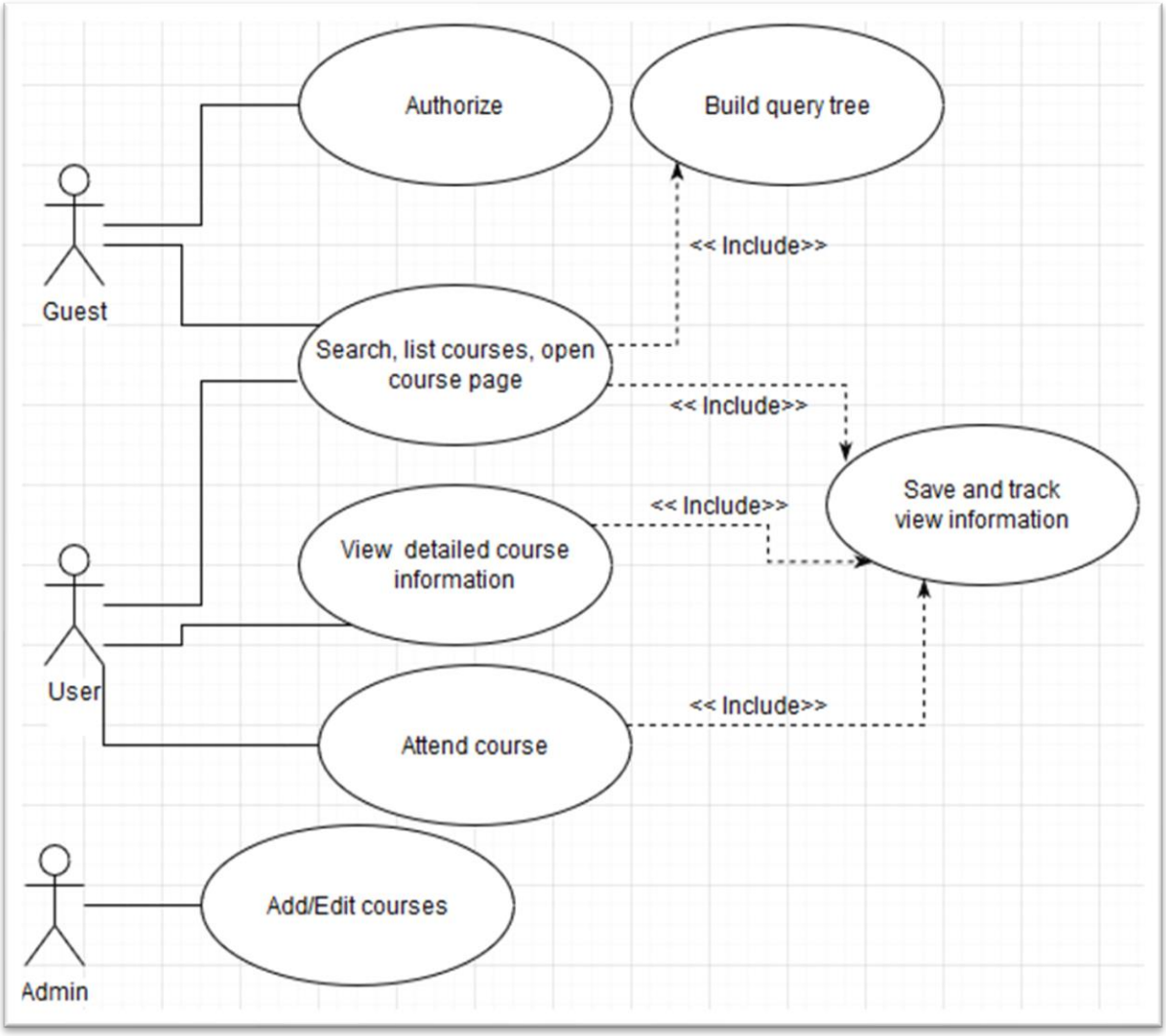

Рисунок 3.1 – Варіанти використання

Тепер варто пояснити, що на ній зображено простим язиком. Тут є ролі:

- гість – це користувач з моменту першого контакту й доти, доки він анонімізований, він може деанонімізуватись, зайдя на відповідні сторінки;

- користувач – це вже наступний етап еволюції людини, як потребітєля системи. Він може проглядати курси, проходити їх, дивитись детальну й повну інформацію про них, його «успіх» про проходження курсів трекається, а, найцікавіше – він отримує персоналізовані рекомендації й вони поступово покращуються;

- адміністратор – це є вже той, хто може додавати, оновлювати й видаляти курси, також він вже може дивитись на репорти з використання системи, які далі можна використовувати для розвитку бізнесу, у якомусь сенсі, це теж є економія часу, бо можна швидко проаналізувати дані, особливо, у тому випадку, якщо вони графічні.

Головним, що робить саме юзер – це є робота з курсами, така, як їх пошук, перегляд, проходження. Пошук курсів за допомогою фільтрів чи пошуку є невід'ємною складовою кожної системи, а такий пошук можна поглиблювати, а кількість фільтрів розширювати. Далі, коли юзер вивча курс, тобто проходе його, то система автоматично ставить галочку на всіх пройдених уроках, так, прогрес буде відмічено, й можна зрозуміти, які частини проекту були опрацьовані. До того ж, це перегукується з можливістю покращення у порівнянні з аналогами, котрі забувають про таку базову функцію. Йде активне синкування з бекендом, щоб все зберіглося, здебільшого, це проходить у фоні, що ще підвищує ступінь юзер експіріенса. Ну, про це не було згадано, але це, звичайно, мається на увазі – перегляд курсів у графі рекомендацій. Це є невід'ємною частиною проекту, до того ж, коли людина щось побачить у цій графі, то вона економить свій час на пошуці й це, знову ж таки, вирішує проблему грамотного використання й оптимізації часу. Згодом, коли буде на меті перехід до комерційної складової, планується ще розширення системи новими фічами, щоб можна було покласти курс до списка бажань, або закладку зробити на ньому, а це, ще більше поліпшить юзер експіріенс.

Далі, перейшовши від головної ролі у системі, до вторинної, варто трохи слів сказати про гостя. Це є така людина, такий користувач, який є невідомим з моменту першого контакту й доти, доки він продовжує бути анонімізованим. Коли він захоче, то може вільно бути деанонімізованим. Для цього йому нужно перейти на сторінку реєстрації, чи, якщо це користувач, що тимчасово перебував у ролі гостя, то на сторінку авторизації.

Ну, й наостанок, варто обозначити останнього за порядком, але не за значенням адміна. Він, немає порядку важливості, бо є невід'ємною частиною функціонування системи, є деяким кором. Він може додавати, оновлювати й

видаляти курси, також він вже може дивитись на репорти з використання системи, які далі можна використовувати для розвитку бізнесу, у якомусь сенсі, це теж є економія часу, бо можна швидко проаналізувати дані, особливо, у тому випадку, якщо вони графічні.

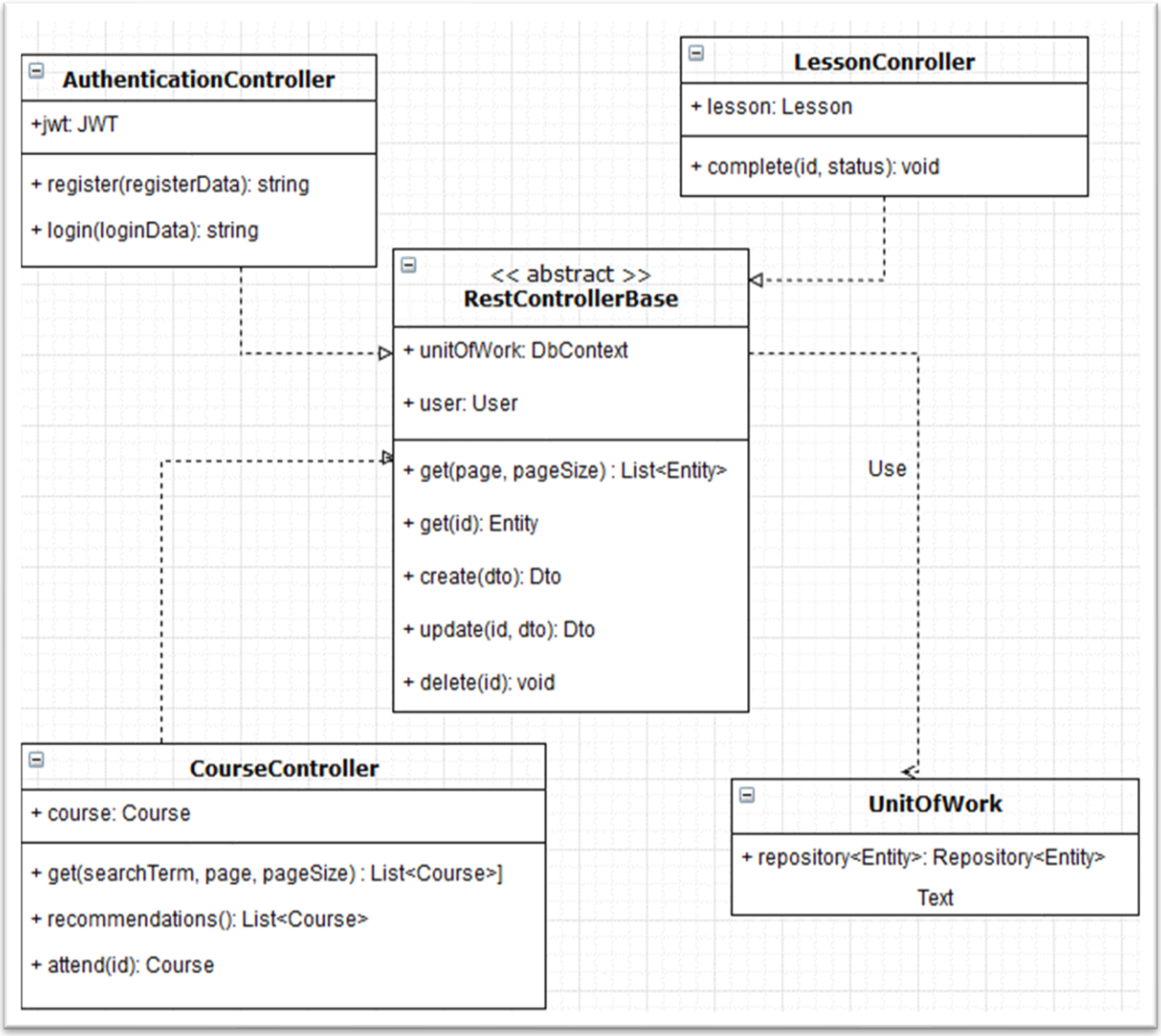

Рисунок 3.2 – Використання паттерну Unit of work

Варто відмітити, що у системі є безліч класів, а на діаграмі варіантів використання було показано найголовніші з них, без зайвих деталей. Зі сторони бекенду було створено наступні модулі: аутентифікаційний, юзерський, інтаграційний, категоріційний, рекомендаційний. А під останньою частиною приховано велику кількість логіки по роботі з базою даних, за допомогою Entity

Framework Core, для обгортки, яка була створена, над нею. Це приклад винаходливості, бо це дуже зручний патерн, який вимагає багато сил для реалізації, його у використання можна побачити на рисунку 3.2, там зображена діаграма класів.

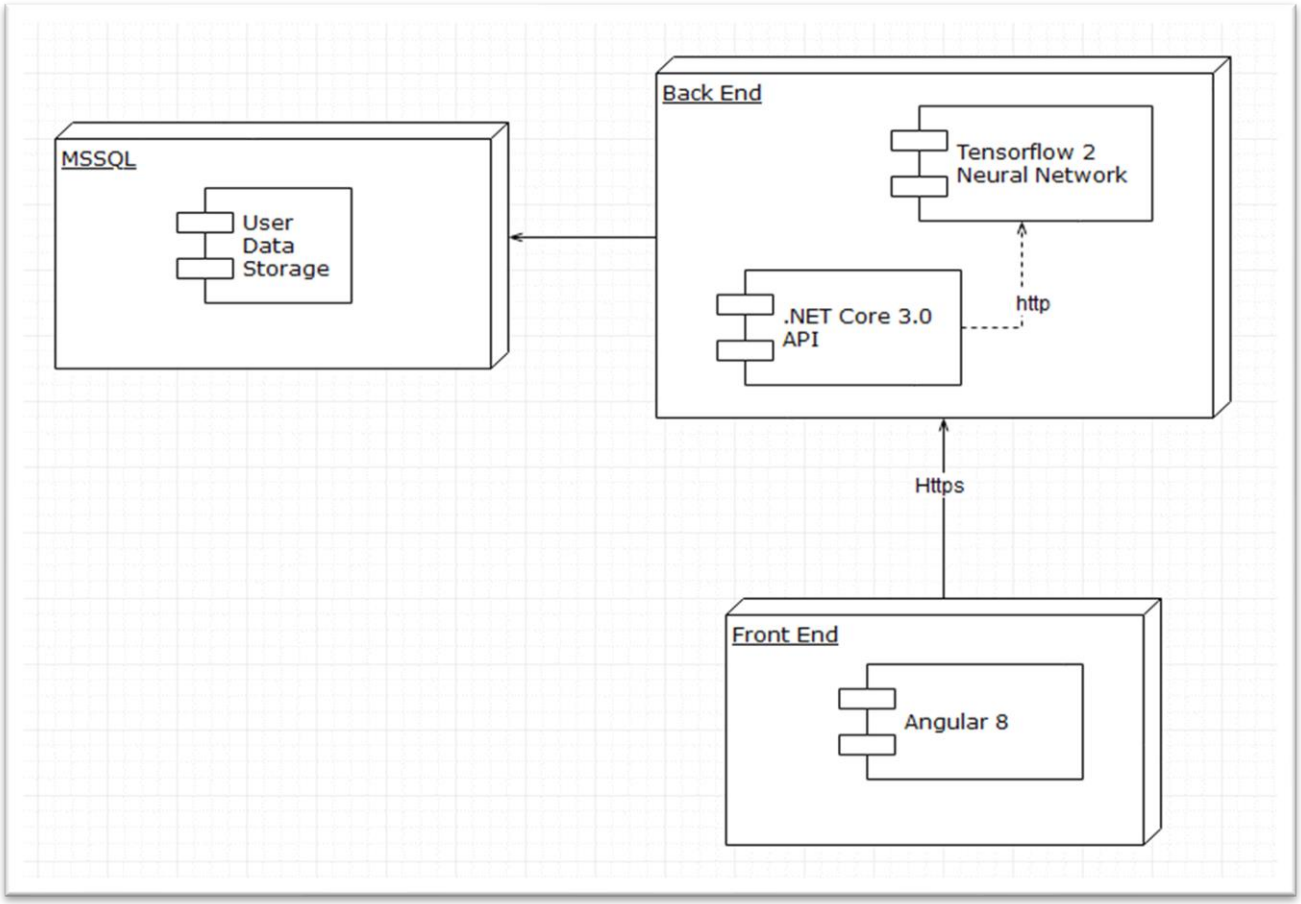

Рисунок 3.3 – Розгортання діаграма

Кілька слів про роботу з базою даних, вона не є мануальною, бо було використано СУБД Entity Framework Core, котра працює з MS SQL. Цей модуль предоставляє базовий клас DbContext, який використовує паттерн білдер для кастомізації, й розширення за допомогою ООП методів, таких, як, наприклад, наслідування.

Наостанок у цьому підрозділі, варто зазначити останню діаграму – розгортання, вона демонструє, що система складена з фронтенду, який можна відкрити через брандмаузер з телефона, комп’ютера, чи планшета. Бекенд частини

– тут власне сам бекенд й рекомендаційний модуль. Ще одна частина, яка перший час буде разгорнута разом з API задля економії ресурсів комп’ютера – це база даних. Звичайне MS SQL сховище даних. І саме цю схему можна побачити на рисунку 3.3.

Трохи подивившись на цю діаграму видно, що фронтенд створений за використання феймворку Angular. Нейронна мереж, тобто рекомендаційний модуль використовую мову Python, а саме фреймворк, неймовірно вдалий, сучасний, популярний фреймворк TensorFlow другої версії. Переходячи до бекенду, він написаний на мові C#, на платформі .NET Core 3. Усі інтеракції використовують xhr реквести між собою, як стандарт для сучасності.

Тепер можна перейти до інструментарію.

3.3 Архітектура через призму інструментарію

Було проаналізовано сучасні рекомендаційні системи на сайтах для навчання, дуже впало у вічі те, що рекомендації іноді доволі банальні. Так, провівши більш детальне дослідження, значний аналіз, де було влкдано багато безцінних сил, було помічено наступне, багацько цих рекомендацій, котрі дають аналоги, базуються на популярності курсу, свіжості курсу, тобто рахується час, як довго ці навчальні посібники є доступними. Ось подолання цих обмежень й має закладатись до архітектури застосунку, яка зараз буде легко пояснена. Систему було розподілено на 4 компоненти, де останні, не за значенням, а у переліку, 3 з них знаходяться на одному сервері. Це зроблено задля економії ресурсів, через фізичні й матеріальні обмеження. Так, бекенд, саме API, нейронний модуль й база даних є розположені на одній стороні й являють собою цільний кластер. Це зручно для пришвидшення реквестів між ними, бо розподілені системи будуть витрачати більше часу на комунікації через значне збільшення комунікаційних вузлів. А з іншого боку – це клієнтський фронтенд додаток, котра має в собі найкращі ідеї з юзер експіріенсу крізь призму людини без уяви. Щоб гість, на якому потенційно можна заробляти гроші, міг взаємодіяти з сайтом йому потрібен стабільний інтернет, бо додавання оффлайн моду можливо лише у подальших версіях. Сама взаємодія має відбуватись

з фронтендом застосунку, увесь доступ до API напряму є неприйнятним, а будь-яка спроба це зробити – йде на страх та розсуд кожного окремого користувача, де він ризикує зламати якусь частину свого експіріенсу за необережністю. Хоч, система й має захист від дурня, але це є таким обмеженням, щоб не підтримувати невігласів у їх скаргах.

Було зроблено клієнт-серверну архітектуру, що є багаторівневою, за використанням паттерну MVC й багатошарову, бо є кілька слоїв – саме фронтенд та бекенд системи. Вона може легко бути масштабовна, відновлювана, підтримана, завдяки своєму, кращому ніж у інших, підходу.

Тепер поглянемо на фронтенд фреймворки.

3.3.1 Порівняльний аналіз фронтенд фреймворків

Багацько різних фреймворків є для того, щоб робити фронтенд. Будуть розглянуті основні 3, одразу ж пропускаючи повз ванільний js, бо він, як відомо, немає жодної архітектури чи структури для побудови великих додатків, а самому ізобритати велосипед не є ефективним.

Було внесено дані до таблиці 3.1, там є вся важлива для вибору інформація, котра стосується цих фреймворків.

Таблиця 3.1 – Потенційні фреймворки для фронтенду

Назва	Архітектура	Доля ринку, %	Документація	Модулі, тис
React	Vanilla	37	-+	250
Angular	MVVM	44	+	235
Vue	MVC	15	+-	145

Фаворит – це є ангуляр, бо він доволі великий й дозволяє зручно працювати з велики проектами, коїм, цей потенційно може стати. Отже, дивлюсь на ангуляр, реакт та вью – всі вони побудовані на різних принципах, різні архітектурні рішення мають під собою й так далі. Варто одразу ж оцінювати й кількість ресурсів для роботи та бібліотек, разом з обширністю ком’юніті.

Загалом, англуяр є переможцем по більшості пунктів, тому його й було обрано. Хоча реакт йде з ним дуже близько, якби не документація, то можна було б линути й до нього. Вью має більш чітку документацію, але він має недостатню популярність, відповідно, складніше знайти потрібну інформацію й менше шанс, що він буде довго жити. Ще, варто сказати, що у кожного з цих фреймворків є можливість працювати з сховищем, хоча першим паттерн redux було використано саме у реакті, де є бібліотека, котра так і називається, але можна використати бібліотеку ngrx у ангулярі, що гарно інтегрована до архітектурного підходу роботи з підписками. А ще зазначу, що тайпскрипт значно покращить підтримуваність системи на більш пізніх стадіях роботи з нею, бо надасть можливість швидше орієнтуватись у коді.

Тепер поглянемо на бекенд фреймворки.

3.3.2 Порівняльний аналіз бекенд фреймворків

Тут також було з чого обирати та над чим розмислювати, основними суперниками були нода, пайтон, а саме – джанго, та асп дот нет. Власне, вони завжди йдуть біч о біч та суперничають, у кожного є переваги й недоліки, але зараз вони будуть порівняні, та буде обрано на основі чесного порівняння. Так, було зроблено таблицю 3.2, у якій і зроблено це порівняння.

Таблиця 3.2 – Порівняння систем для розробки серверу

Назва	Юзерів, тис.	Архітектура	Масштабовна	Бібліотеки, тис.
Node.js	450	-	Ні	231
ASP.NET	340	MVC	Так	120
Django	205	MVVM	Ні	170

Так, дивлячись на кількість юзерів, можна сказати, що перемагає нода, яка є явним фаворитом, бо джанго поступається у 2 рази, а асп знаходиться у середині, але в даному випадку все ж переважає, бо комьюніті достатньо велика й Microsoft займаються підтримкою, документуванням ASP.NET, тому він тут у фаворитах з

першої колонки. Далі, дивлюсь на архітектуру й бачу, що надо не має принципового паттерну під собою, тому її одразу скіпну, бо це на пізніших стадіях розробки може вилезти боком й завадить масштабуванню, яке, до речі, вона має слабке. До джангу и .NET немає ніяких претензій, ці паттерни близькі, майже взаємозамінні й рівноцінні. Теперь дивлюсь саме на масштабування, тут вже фаворит підкреслюється знову, навіть коментувати нема чого, конкуренти програють всуху по цьому пункту. Тепер дивлюсь на кількість модулів, .NET вже сильно поступається, але всі інші пункти занадто сильно підкреслюють сильні сторони його, тому все ж така поразка у модулях не є вирішальною. До того ж, ці модулі при такий великій кількості не є вирішальними. Отже, тут доволі легко помітити, що ASP.NET перемагає, саме на цьому фреймворку й було розроблено всю систему. До того ж, є така штука, така СУБД як Entity Framework версії Core, тому це ще добавляє бажання обрати саме цю технологію. Отже, явний фаворит був обраний.

Тепер поглянемо на інструментарій для написання нейронних модулей.

3.3.3 Порівняльний аналіз модулів для побудови нейронних моделей

Переходячи до вибору технології для створення модулю надання рекомендацій. Так, перший кандидат – це TensorFlow, який розроблено Google Brain і він активно використовується в Google як для дослідницьких, так і виробничих потреб. А його головний конкурент PyTorch — це двоюрідний брат Torch на основі lua, який був розроблений та використаний у Facebook. Однак PyTorch не є простим набором обгорток для підтримки популярної мови, він був переписаний і адаптований, щоб він був швидким і був рідним.

Що ж, огляд буде розпочато й пояснено з порівняння цих технологій з кінця. Так, виходить, що у них є незначна різниця у кількості модулей, що не є критичною проблемою ні для жодного з них, але якби вони були, то це було б плюсом. Так, така модульність в обох випадках є зручною.

Далі, паралелізм, PyTorch є явним фаворитом, бо у нього він нативний й більш зручний, але він притаманний й у кого конкурента, хоч і у менш зручному вигляді – через використання обгорток.

Результати порівняння, наведено у таблиці 3.3.

Таблиця 3.3 – Порівняння нейронних технологій

Назва	Абстракція	Відладка	Паралелізм	Модулі, тис
TensorFlow	Вища	Пошагова	+-	47
PyTorch	Середня	Консольна	+	35

Але у випадку цієї дослідницької роботи це не є вирішальним аргументом. Далі, відладка, тут TensorFlow явно перемагає і це вже явне зазіхання, щоб він був обраний, бо зручність відладки це є все під час такої аналітичної роботи. Але не варто зкидати його конкурента одразу, варто дати йому шанс, йдемо далі. Тут йде найсмачніша частина – рівень абстракції, виходить, що можна абстрагуватись да найвищого рівня у TensorFlow, так, це дуже добрий знак. Це означає, що не обов’язково будувати нізкорівневі математичні розрахунки, а можна їх взяти з коробки. Це однозначно виділяє переможця, тому саме TensorFlow й було використано.

Далі, розглянемо найцікавіші методи.

3.4 Приклади найцікавіших алгоритмів та методів

Вхідними даними для отримання рекомендації є час, який людина провела на курсах, сумарно на всіх лекціях й описах курсу. Така інформація є потрібною, бо варто від чогось йти, коли створюється прогноз, у моєму кейсі – рекомендація. Ще одним параметром є проставлені вподобайки на курси й конкретні уроки, це дозволить ще розширити профіль користувача, з ціллю покращення рекомендацій. Для того, щоб навчити нейронну мережу думати, варто десь взяти дані, ці дані є замоканими і напряму витягуються з бази даних системи, поступово таке рішення

можна буде масштабувати чи модернізувати, але в даному випадку цього достатньо, за умови відповідності всім лімітам системи.

Ітеративно, коли накопичується деяка статистика на сайті, тобто люди проходять й відвідують більше курсів, ставлять свої лайки, вподобайки, на курси й конкретні уроки, з'являється більше інформації, за якою можна будувати нейронну мережу, під будуванням мається на увазі в тому числі й перенавчання, оновлення нейронних зв'язків іншими вагами, що дасть точніші рекомендації. Трохи поглиблюючись у технічну частину, важливу роль відіграє модуль Vectori, який трансформує вхідні дані до відповідного вектору, а цей вектор використовується як під час навчання, так й при наданні саме рекомендацій. Як було зазначено раніше, час та вподобайки конвертуються до відповідних векторів й вони далі використовуються у тому форматі, який був зазначений вище. Що є приємним у роботі з нейронною мережу – це те, що можна різноманітним чином модернізувати її, змінювати деякі сетинги, це дозволить протестувати гіпотези щодо збільшення точності надання прогнозу. Одним з шумів, котрі є деяким лімітом системи, є наступна проблема, невідомо, чи юзер справді був зацікавлений у курсі, чи він забува закрити сторінку коли раптово клікнув. Але це є очікуваним обмеженням і зазвичай користувачі не притаманні надовго лишати вкладки, тому цей варіант і був обраний, як ключова метрика й емпіричні досліди довели, що це не дарма. Подитожу, що загалом було доведено, що ця статистична похибка має вплив, але занадто малий, щоб з нею серйозно рахуватись. Та, чим більше даних статистичних буде у системі, тим краще такі шуми можуть бути виявлені й придушені. Але це буде вже згодом.

Варто ще зауважити, чому такий підхід був обраний, він був обраний тому шо у багатьох соревнуваннях такі підходи отримували призові місця, й про це було багато інформації, що у роботі дуже допомогло у подальшому розвитку цієї теми. Але однією зі складностей було порівняння даних з іншими системами й тестування алгоритму, адже комерційна таємниця крупних проектов забороняє їм надавати подробиці роботи власних систем.

4 РЕАЛІЗАЦІЯ

4.1 Бекенд

На основі вибору технологій зрозуміло, чому саме цей бекенд фреймворк й було обрано, бо він найкраще відповідає всім потребам. Так, бекенд було розроблено на платформі .NET версії Core, де сам бекенд було написано використовуючи конкретний модуль – ASP.NET Core у цій платформі.

За підход було обрано рестфул апі, це підход, який містить у собі певну стандартизацію всіх ендпоінтів до формату, коли ми явно вказуємо модуль у урлі, інакше модуль ще можна називати контроллером, далі за ним йде назва конкретного еквшена. Самого екшена може й не бути, це буде позначати CRUD операції, базові. Доступ до них відбувається за допомогою відповідних HTTP реквестів. Так, для того щоб отримати дані – ми маємо створити GET реквест, у якому в урлі є 2 варіанти як кверити сутності. Перший – це коли ми нічого не пишемо, лише гет у конкретному контроллері – тут ми отримуємо список, а коли передаємо строку пошуку, фільтри з пагінацією, то можемо ще й відповідно шукати, фільтрувати та пагінувати крізь дані. І саме цей підхід й було заімплементовано для роботи з головною сторінкою додатку, де є саме вказані фільтр, пагінація та пошук. Другий варіант – це передати у адресі id шуканої сутності, тут вона одразу ж й буде знайдена й повернута за допомогою xhr до відправника. Далі, якщо ми хочемо зберегти якусь сутність, то робимо вже POST реквест, наприклад, на той самий CoureController – так ми можемо створювати курси, але це доступно лише для адміну, що й регулюється атрибутами авторизаціонними. Далі, адмін ще має вміти й редагувати сутності – за таку дію відповідає наступний тип реквесту. Це PUT реквест, він за своєю структурою є таким само, як й його попередник – пост. Але різниця полягає у тому, що можна передати у урлу, по аналогії з гетом, ще й id сутності, яка буде редагуватись. І останній, але не по важливості – це DELETE тип реквесту, за допомогою нього адміністратор може видалити курс, якщо, наприклад, викладач так попросить, бо така ситуація може виникнути.

Всі ці контролери мають бути якось згруповані, а саме вони згруповані так, як зображено на рисунку 4.1

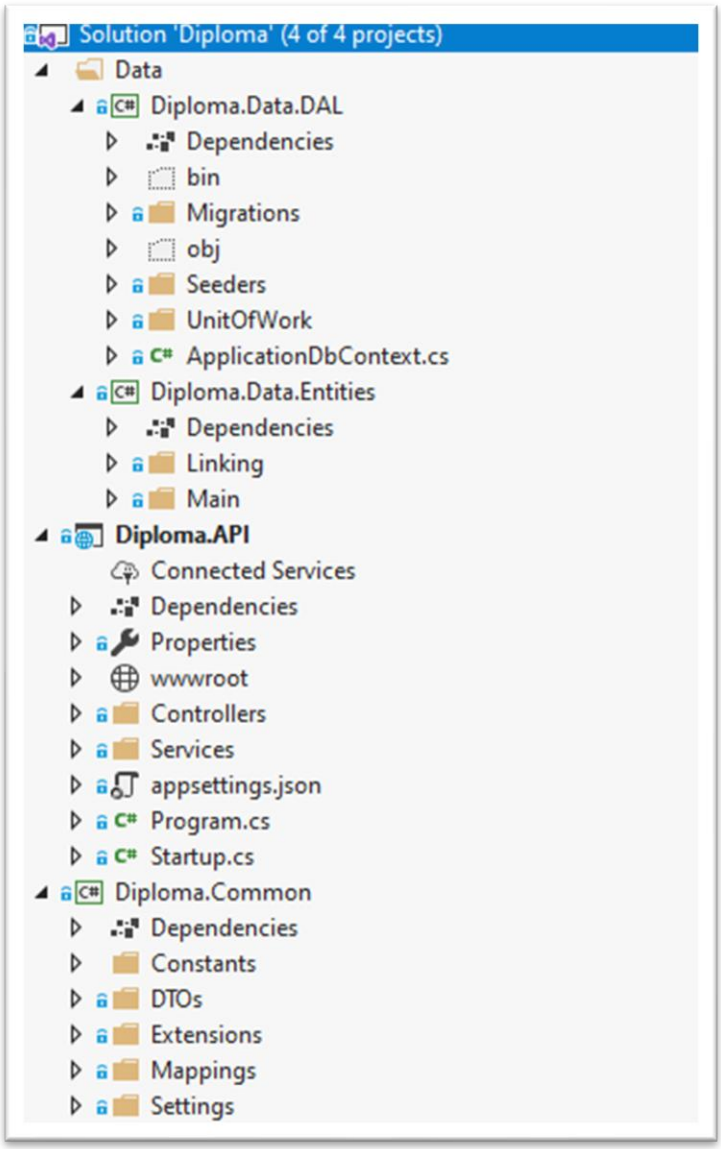

Рисунок 4.1 – Бекенд архітектура

Загалом є декілька важливих частин солюшину – це проект Common, який слугує містилищем для загальних класів, наприклад, для dto чи констант. Далі, йде пара DAL та Entites – вони відповідають за доступ до бази даних та інкапсуляцію її сутностей, так, щоб вони не витікали назовні й кінцевий клієнт не знав, що саме за структура бази запроваджена. Це робиться для того, щоб шахраї не могли змогу напряму пробувати модифікувати базу, таблиці. Тобто це зроблено не лише для

зручності, а для безпеки самого застосунку. Ну і останній проект – це власне API, який й використовується для комунікації з клієнтом, у ньому тимчасово знаходиться й бізнес логіка застосунку.

Тепер перейдемо до рекомендаційного блоку.

4.2 Рекомендаційний блок

Маючи повну теоретичну базу, яку було розписано вище, навівши приклади формул, назви алгоритмів й загальну ідею їх використання, розуміння того, який фреймворк використовувати, було розшукано правильний ресурс для розгортання нейронної мережі та зручного способу роботи з нею, так, було знайдено Jupiter Notebook. Ця штука корисна, бо дозволяє інтерактивно взаємодіяти на етапі розробки, та досить легко відкривати доступ ззовні за допомогою модульності, через всім відомий HTTP протокол, його секьюрну версію HTTPS.

Так, за розробки додатку, було написано наступні методи, які маплять вхідні дані у вектор, котрий потім використовується як при навчанні нейронної мережі, так й при створенні прогнозу:

```
like_embedding = Embedding(
     name = 'like_embedding',
     input_dim = len(like_index),
     output_dim = embedding_size
)(like)
time_spent_embedding = Embedding(
     name = 'time_spent_embedding',
     input_dim = len(time_spent_index),
     output_dim = embedding_size
)(time_spent)
```

Цей код обробляє 2 вхідних масива даних, де обидва презентують необіхдні для рекомендацій дані. Так, перший містить векторювання списка лайків, тут йде його перетворення через інструменти TensorFlow до вектору, котрий буде використовуватись далі. Та видно, що такий вектор не один, їх два, другий збудовано з часу, який юзер витратив, переглядаючи курси. Цей масив дуже

швидко і легко перетворюється на вектор. Тепер, можемо змержити ці 2 вектори до одного – було використовую для цього наступний код:

```
merged = Dot(
     name = 'dot_product',
     normalize = True,
     axes = 2
)([like_embedding, time_spent_embedding])
merged = Reshape(target_shape = [1])(merged)
merged = Dense(1, activation = 'sigmoid')(merged)
```

Як можна зрозуміти, на вхід йде двовимірний масив, де 1-й вимір – список, а 2-й – це вектора. Так, axes позначає розмірність вхідних даних, далі вони за допомогою вбудованих до бібліотеки методів оброблюються, щоб отримати одномірний ваговий вектор. Ну й наступним йде створення базової одношарової моделі, на яку трохи згодом буде додана основна частина моделі, так, маємо наступний код:

```
model = Model(
     inputs = [like, time_spent],
     outputs = merged
)
model.compile(
     optimizer = 'Adam',
     loss = 'binary_crossentropy',
     metrics = ['accuracy']
)
```

Як щойно було зазначено, це є вже створення першої частини моделі – її базу, на яку й далі було додано шари по обробці. Як зрозуміло з назви – нейронна мережа, це мережа нейронів, де всі шари, а зверху лише база, накладаються один на одний. Всього вийшло 4 основних шари у цій нейронній мережі, а їх математичне обгрунтування є у відповідному розділі, та 3 допоміжні, один з яких й є наведеним вище. Так, поступово дані проходять крізь мережу, яка по своїй суті схожа на павутиння, де кожен попередній рівень пов’язаний з наступним, а всі нейрони злінковані між собою й мають, можна так сказати, транзитивну залежність, бо зміна вагів в одному місці явно змусить мережу перерахувати ваги

й у інших рівнях, але це працює в напрямку від початку до кінця, а не з вихідної частини моделі до початкової.

Тепер перейдемо до фронтенду.

4.3 Фронтенд

Фінальним етапом було зробити фронтенд додатку. Він є чи не найважливішим, бо якою б доброю й економящею час система не була, якщо буде погана архітектура, чи оформлення (про яку буде розказано трохи далі), ні користувач не зможе працювати з додатком, ні розробник не зможе. Так, за використання технології Angular було віддзеркалено бекенд-модулі у фронтенд частині системи.

Це було зроблено задля того, щоб під час розробки було більш зручніше працювати з усією системою. Такий підхід до модульності значно полегшує витрати сил на процес розробки й адаптації. А коли команда буде розширятись, то це зменшить тривалість процесу адаптації.

З приводу розробки, лише варто відмітити, що спочатку було створено бекенд й модуль рекомендацій, а вже потім клієнт, бо інакше можна виявити невідповідність даних на більш пізніх етапах розробки, та буде складно проходити процес планування архітектури застосунки, бо формат зберігання даних не завжди є тривіальним, хоча такий підхід іноді може мати місце.

Якщо опускати основні модулі, до них можна повернусь згодом, бо загальна ідея зрозуміла. Так, було створено 2 базисні методи – core та shared. Вони містять у собі 2 суміжні, але все ж різні групи файлів, так shared містить у собі компоненти, хелпери, глобальні моделі, налаштування. А core – основні компоненти системи, такі як хедер та футер, сінглтон сервіси та авторизаційні налаштування.

Сінглтон сервіси – це сервіси, що пронизують систему й існують у її скопі лише в одному варіанті, в одному інстансі, такі, наприклад, як logger чи http сервіси у системі.

Далі, знаходяться вже модулі самої системи – від модуля доступного анонімному користувачу, до адміна. Те, хто може зайти до якого модуля розрулюється за допомогою гардів, які лежать у core модулі.

Так, структуру фронтенд проекту зображено на рисунку 4.2.

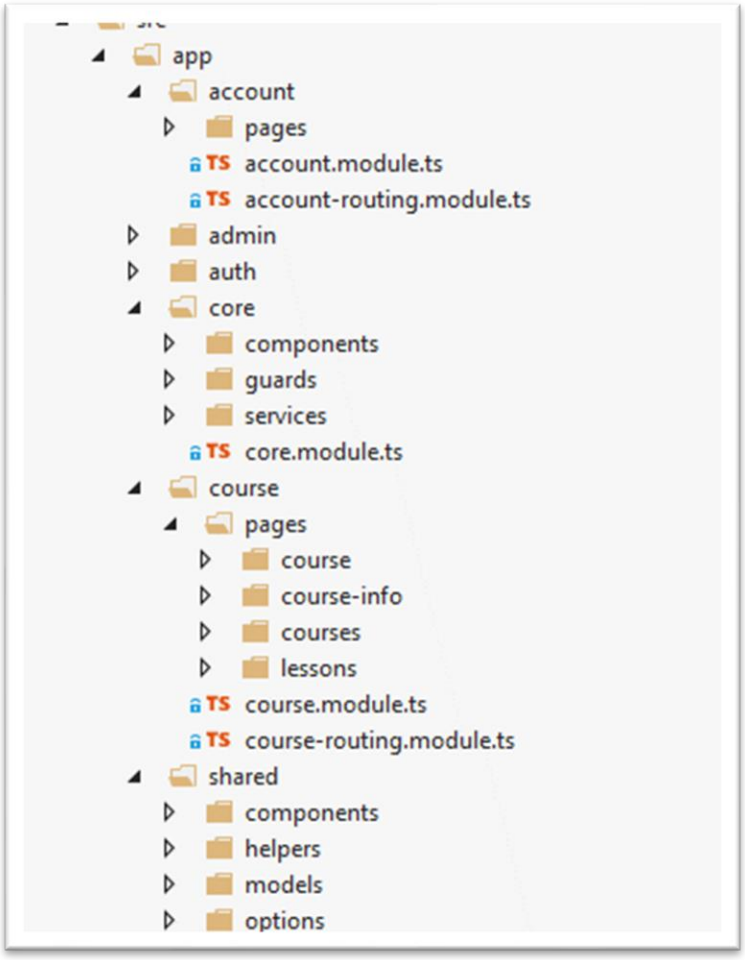

Рисунок 4.2 – Модулі фронтенд частини

Серед каталогу сервісів є логування, яке потрібно для виявлення проблем чи аналізу, підбивання статистики, ще є HttpService, який дає змогу комунікувати з бекендом через xhr запити, шляхом використання REST API бекенду. І загалом, такі системи вимагають менше документації, менше сил для підтримки, бо їх структура зрозуміла й зручна, з нею легко працювати.

Нижче, наведено кілька прикладів підходу, до написання коду, так, нижче наведено метод реєстрації, який було використано у додатку та модель даних, сам підхід з кешуванням:

```
Register = (dto: RegisterDto) => ({
     httpClient
          .doPost<Account>(
               `${apiUrl}/authentication/reg`,
               dto
     ).cache(account);
})
```

Загалом, це така структура написання реквестів, де передається тип dto до методу, а далі використовую внутрішній httpClient ангуляра, щоб передати xhr реквест до сервера. Ще варто зазначити метод cache, який дозволить не виконувати повторно цей реквест, а взяти з кешу дані, тут цей метод зображен для наглядності, але він активно використовується на більшість реквестів, де це є необхідним. До того ж, варто відмітити, що є 2 етапи валідації – перший на клієнті, другий – на сервері. Так, система уникає зайвого навантаження на бекенд частину системи й додатково дає змогу для API потенційно стати відкритим, для публічного використання. В роботі використовуються всі основні типи http реквестів – гет, пост, пут, деліт. Вони вимагаються потребам REST API підходу, так, можна бути впевненим у сумісності. Ще варто наголосити на тому, що фронтенд додаток повторює зовнішню структуру моделей бекенду, що посилює можливість швидшої інтеграції та більш легкого інтуітивного розуміння системи, котра може бути легко масштабована завдяки цьому.

Тепер варто розглянути інтерфейс.

4.4 Інтерфейс

Юзер інтерфейс, чи, інакше кажучи, юзерський інтерфейс є надважливою складовою будь-якої публічної системи. Та навіть не лише публічної, кожної, бо іноді можна стикнутись з внутрішніми системами, що мають жахливий вигляд і це заважатиме концентруватись на роботі, а якщо UX був також поганим, то кількість

витраченого на роботу часу зросте майже вдвічі. Так, важно правильно підібрати кольори – де за основу було взяти синій, приємний оку колір, який є основним у стилістиці bootstrap. А, як відомо, такі модулі, гіганти у цій справі, вміють робити правильні речі. Отже, синій, такий, небесно-блакитний був взятий за основу. Також, шрифт є важливою складовою, ось, наприклад, якщо шрифти незручно читати, ті клієнт радше перейде до іншої системи, аніж буде розбиратись та звикати.

Переходячи, до конкретних скрінів, йдемо по їх логічному розташуванню. Так, скрін реєстрації, яка є обов'язковою для повноцінного користування системою зображено на рисунку 3.4.

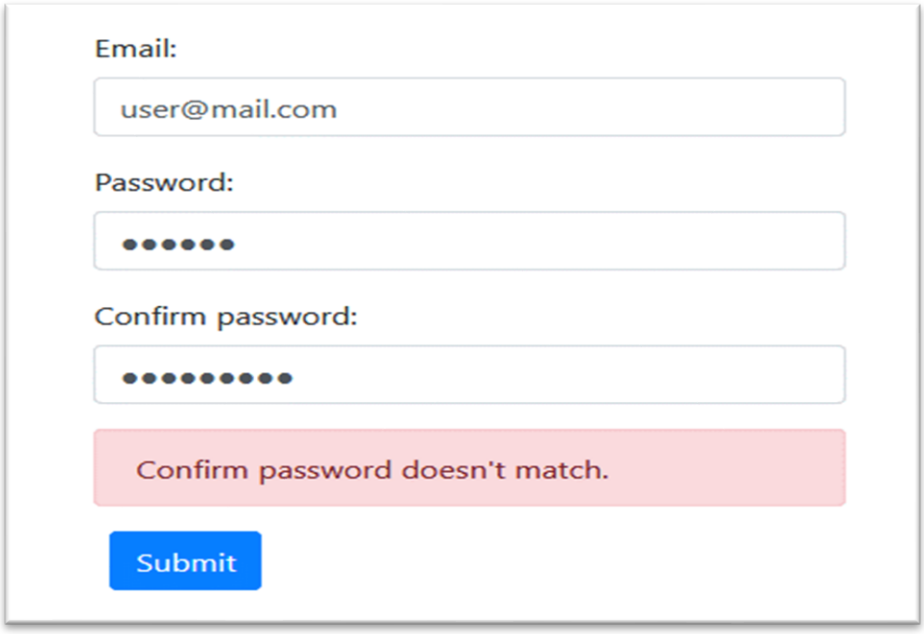

Рисунок 3.4 – Скрін з реєстрацією

Тут можна бачити, що валідація також не була забута, це убереже систему від проблем, які потенційно могли б виникнути у майбутньому з форматом даних юзера. Ну, а після реєстрацію можна авторизуватись.

Далі, варто зазначити, що кожна компонента сайту, як кнопка, меню, посилання – є важливими, вони мають гарно виглядати й знаходитись у доречному місці, інакше це може зробити негативний зріст кількості користувачів всієї

системи. Для того, щоб стандартузавти свій підход, було обрано найбільшого гравця на IT-ринку – це Material Design стандарти від Google й максимально було линуто до його базових принципів. Що ж, варто замітити, що всесвітньо розповсюджені ці підходи є.

Тут пропрацьовані всю нюанси – шейди, тіні, хувер та клік ефекти, анімації інтеракцій. Такий інтерактивний підхід до юаю, позитивно впливає на весь плин роботи з сайтом, збільшує інтерес користувача до застосунку. До того ж проводились опроси, щоб виявити який з кількох інтерфейсів на вибір краще вписується у концепцію сайту, так, емпіричним шляхом було визначно багато моментів для покращення.

Для цього також є спеціальна пейджа, зазначу, що на ній також є валідація. Цей чудовий дизайн можна побачити на рисунку 3.5, там є 2 філда – електронна адреса користувача й пароль від його учотного запису.

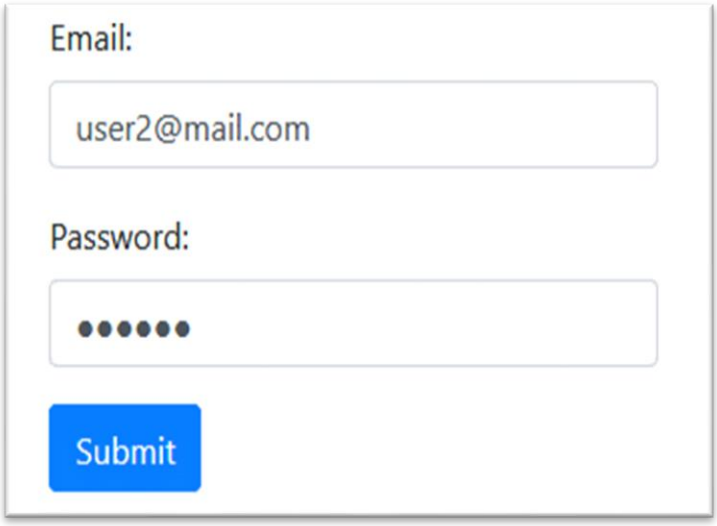

Рисунок 3.5 – Екран авторизації

Після авторизації можна переходити до користуванням додатком. Тут, вже всі профільні дані будуть доступні, а рекомендації підганяються під конкретні потреби користувача, що значно зменшує кількість часу, на пошук конкретної навчальної програми, й, також, зазначу, що навчання проходить онлайн, що, знову

ж таки, зменшить кількість часу на дорогу до навчального закладу, у якому цей матеріал міг би бути начитаний.

Так, екран головної сторінки є на рисунку 3.6.

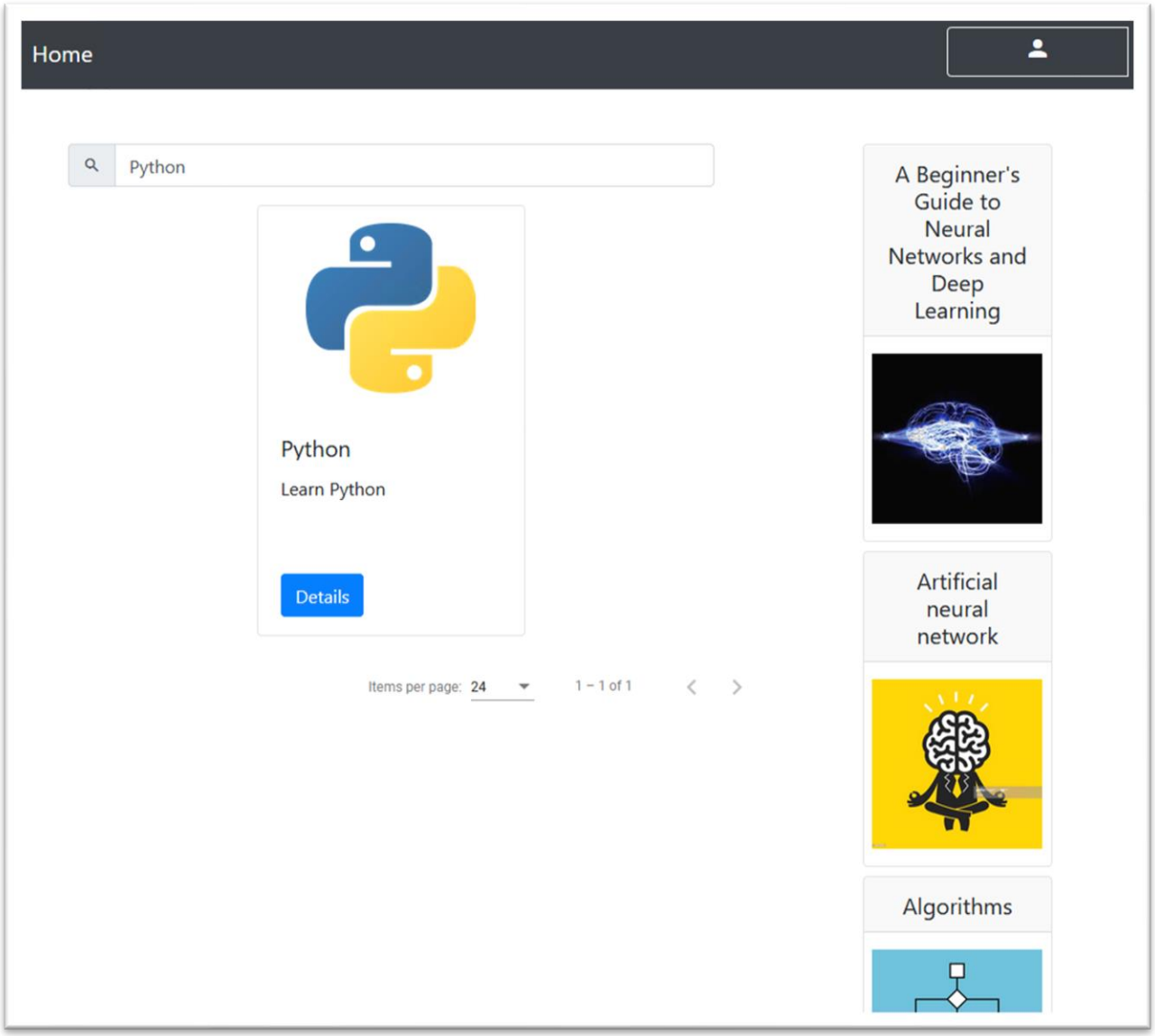

Рисунок 3.6 – Перша пейджа

Тепер, глянемо на основну пейджу додатку, тут є вікно, тобто поле, для пошуку. Якщо вписати ключове слово, то частковим пошуком за використання суфіксного дерева. Далі, фільтрація дає змогу швидко знайти потрібний курс, а фільтри можуть бути додані, прибрані, розширені за вимоги юзера. Ну, й, нарешті, пагінація, що виключно позитивно впливає на юзер експіріенсі, бо вона дає

можливість економити час на очікуванні завантаження сторінки сайту, й, до того ж, можна відкрити потрібну сторінку, щоб не шукати у всьому списку щось ще раз.

Тут, також, трекається прогрес проходження кожного курсу, тому юзер одразу може відкрити те місце, де він припинив навчання минулого разу. Отож, графа рекомендацій значно зменшує час на пошук курсів. Ну ось й було показано головну сторінку, тепер, з неї можна перейти до самого курсу рисунок 3.7.

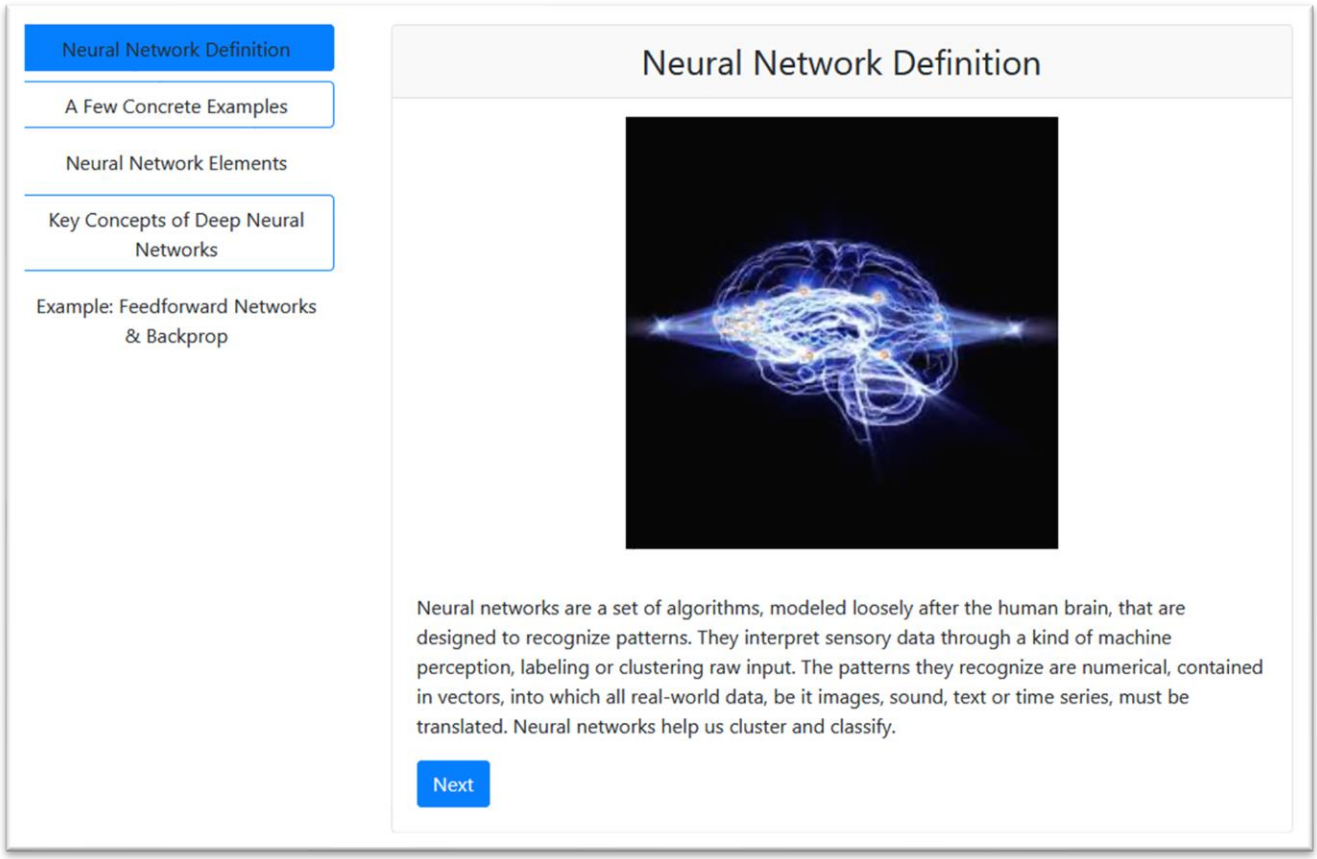

Рисунок 3.7 – Матеріал курсу - лекція

Якщо навести курсор на курс, будь він у графі рекомендацій, що наведена зправа, чи у результат пошуку, й нажати мишкою на курс, то користувача буде переведено на найголовніший модуль, зрозуміло, що не рахуючи модуль рекомендацій, модуль курсів з відповідною сторінку проходження курсу. Основна ідея заключається у створенні такої архітектури, щоб вона могла легко розширюватись, отож були наведені саме перші мокові взаємодії – за допомогою тексту, відео, картинок. Без залучення презентацій, слайдів, інтерактивних дошок,

чату. Ці модернізації можливі у подальшому, коли буде подано фінансування до системи з боку вишу, інвесторів, чи кого завгодно, хто має достатньо грошей, щоб оплатити ці послуги. Так, зупинившись на цій ноті, варто продемонструвати проходження курсу, так, статус проходження лекції є зображеним зліва, а матеріал лекції наведено зправа.

А щоб відкрити список свої курсів, після авторизації треба зайти до свого профілю, там буду графа придбаних курсів, й у ній будуть наведені всі придбані курси. Так, нажавши на один з них можна вже перейти до лекцій, одна з яких й наведена вище. Далі, можна можна подивитись на прогрес проходження зліва від лекцій. І що є приємним бонусом, треба лише перейти до наступної лекції, щоб прогрес було збережено. І, коли людина повторно відкриває курс, то її буде направлено на ту лекції, на якій вона була.

5 ВЕРИФІКАЦІЯ ТА ТЕСТУВАННЯ СИСТЕМИ

5.1 Навантажне

Так як система має пройти повний цикл розробки, то було проведено ще й тестування у повному циклі. Починаючи з навантажної частини – це дозволить дізнатись чи є десь якість неоптимальні підходи до написання. Так, було створено деякий тест, що відправить значну кількість реквестів на кожний окремо, чи у сукупності ендпоінти свого застосунку, чи ще можна зробити інтерактив через натискання ботом кнопок на юаї й відправленні реквестів. За допомогою програмного забезпечення Apache JMeter, що наведена на рисунку 5.1, буде проводено тестування.

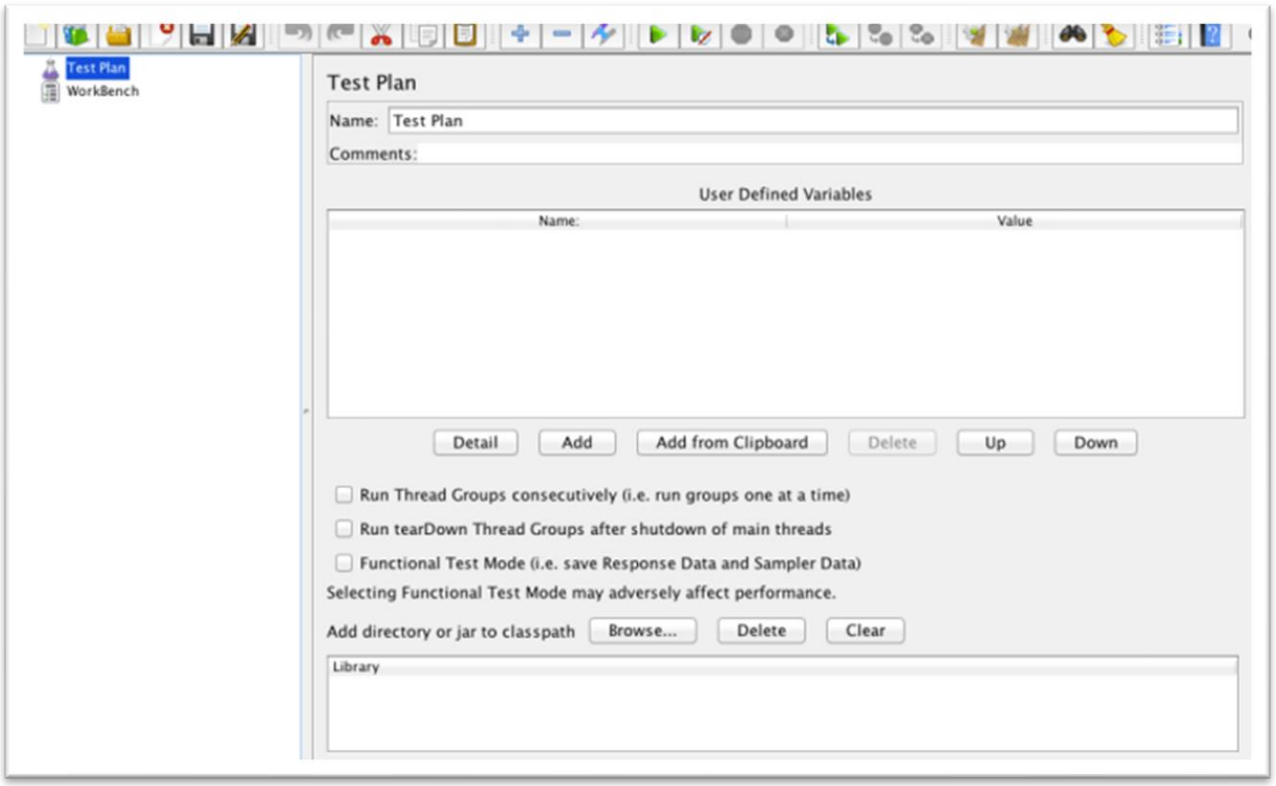

Рисунок 5.1 – Перший тест план

Можна напряму тестувати юай, але це вже буде дещо інший слой тестування, але також можливо ще й так. Але тут було напряму навантажно тестовано саме API та модуль рекомендацій. Так, головним показником є час, за який операцію буде

виконано, тобто, час обробки реквесту. Та кількість таких оброблених запитів, бо якщо система буде неспроможна обробляти одночасно хоча б кілька тисяч реквестів, то це показуватиме її слабкість, потрібно буде замислитись над тим, як можна масштабувати, вертикально – покращити сам бекенд системи й оптимізувати процеси, чи горизонтально – додавати нові кластери та за допомогою лоад балансера, nginx, наприклад, налаштовувати поведінку системи й розподілення реквестів між серверами.

Як видно з скріншоту, було створено перший тестовий план, до якого далі було додано сценарії для тестування. Так, було додано 1000 паралельних сценаріїв, це наче юзер, що буде відправляти запрос. А тест було вказано, що буде проходити протягом 30 секунд, так, була змога точно побачити, чи є вузькі місця у системі. Це буде виглядати, як просадки чи піки. Тепер, буде заповнено сценарій – до нього було вписано по 10 реквестів з тестовими даними. Було додано авторизацію, реєстрацію, отримання рекомендацій, фільтрування, пошук, проходження курсу, деякі з цих реквестів по кілька разів. Тест було запущено, за його результатами було ортимано криву деградаці, яка зображена на рисунку 5.2, на якій можна оцінити роботу системи за допомогою візуальної компоненти.

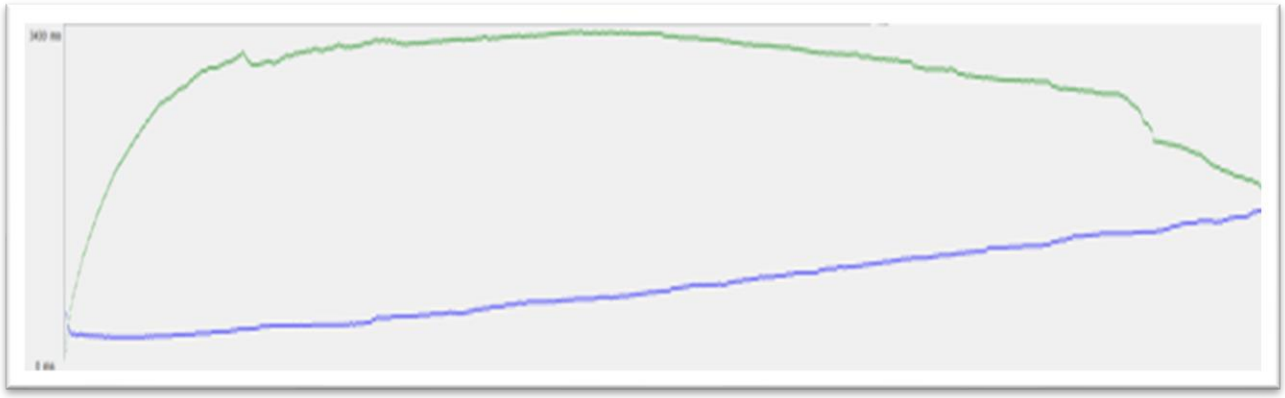

Рисунок 5.2 – Крива деградації

Дивлячись на графік, можна сказати, що тут є дів лінії – зелена та синя, де зелена це скільки реквестів здатна обробити система за одиницю часу, а синя – середній час відгуку на реквест.

Верхня межа графіку – 3.5 секунди, це означає, що це максимально очікуваний час відклику, який зумовлений фізичними здатностями серверу. В той самий час нижня крива поступово зростає, що є гарним показником, бо ніде не нарива. Це означає, що навантаження йде рівномірно й залежить повністю від ресурсів системи. Вона повільно доходить до середини – це 1.71 секунди на надання відповіді у середньому у піку навантаження. Такий показник є добрим, він показує стабільність роботи всього бекенду проекту, а особливо це радує, коли видно, що модуль рекомендацій теж стабільно й чітко працює, що дозволяє без просадок надвати рекомендації..

Тепер перейдемо до мануального тестування.

5.2 Мануальне

Потрібно тестувати всі три частини, якщо дуже бажати, то й всі чотири, але залишимо базу у спокої, й вважатимо, що дані цілісні, бо вони всі замокані й немає причин навмисно туди впихувати невалідні дані. Так, було використано Postman для того, щоб тестувати API системи, він дозволяє зручно форматувати й відправляти відповідні реквести до API частини додатку, а таке форматування й обробa реквестів значно пришвидшуюь процес розробки.

За допомогою постмена можна провести повне тестування всього бекенду застосунку. А ще більше радує можливість зберігання усіх тест-кейсів у так званих колекціях. Надалі ці колекції можна буде інтегрувати у навантажне тестування, чи використати під час написання автотестів [13]. Сам постмен є широкорозповсюдженим та має широкий спектр аналітичного інструментарію, який за потреби можна використати. Ще одна приємна особливість – це наявність інтеграцій, що перекликається з тим, що було написано вище, про можливість створення автоматичних тестів.

Весь процес тестування серверної частини проходим за допомогою надсилання валідних й не валідних даних до серверу крізь http протокол, так було спочатку створено тест-кейси, котрі описані у джирі, далі, проходячись по дошці,

були дістаті ці кейси й пропрацьовані різні частини системи. Особливої пильності було приділено саме рекомендаційному модулю.

Приклад такого тестування наведено на рисунку 5.2.

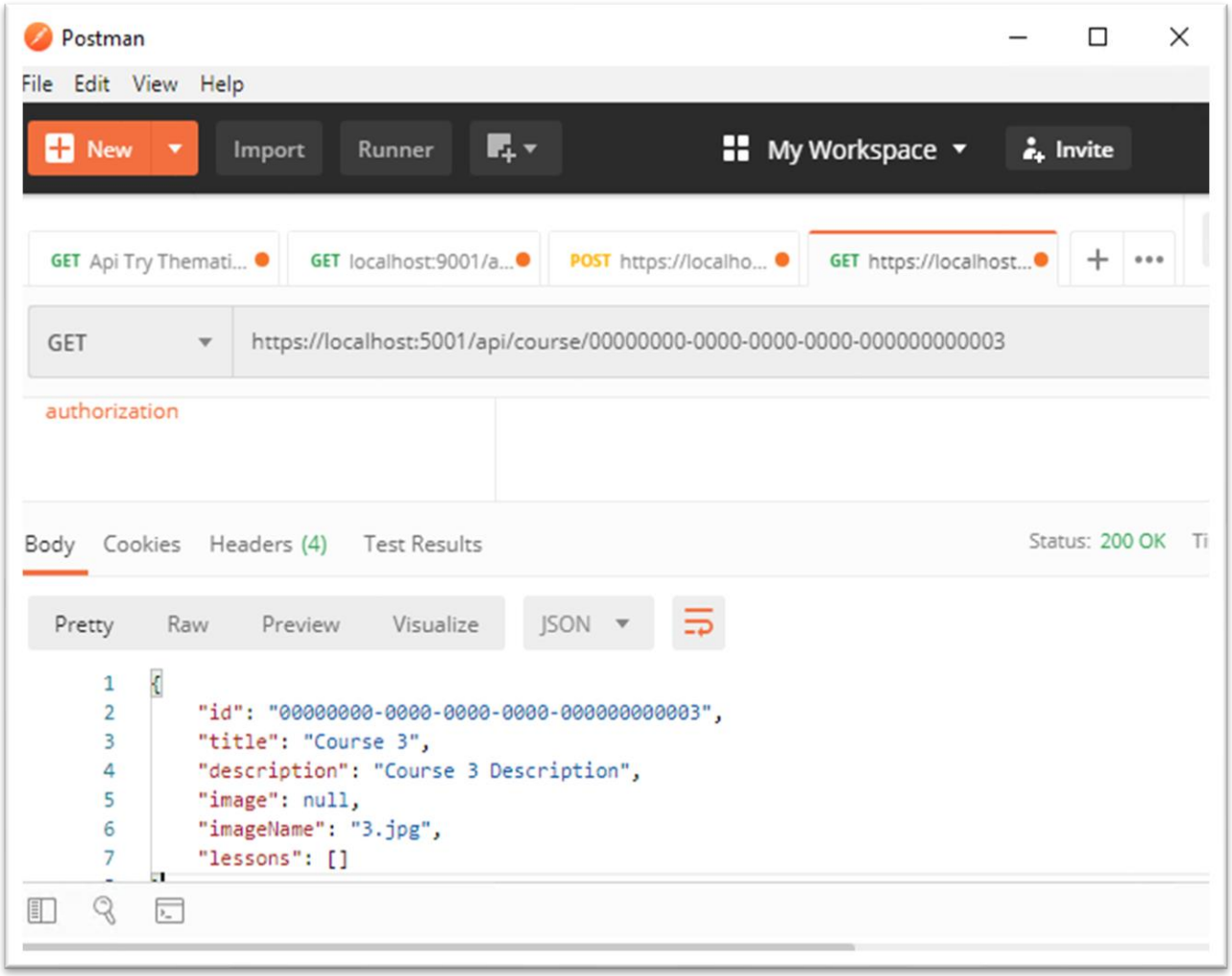

Рисунок 5.2 – Процес тестування бекенду

Так, було виявлено кілька дефектів, що пов’язані з неспроможністю системи парсити деякі символи, котрі було заблокоано за допомогою валідації, також було виявлено кілька ендпоінтів, що не були дороблені й кидали помилки за спроби їх кверити, що також були виправлені. Так, цей етап пройшов успішно й продуктивно.

Тепер, протестуємо точність рекомендацій.

5.3 Тестування точності рекомендації

Протягом розробки модуля для надання рекомендацій, як було зазначено раніше, було використано обмежені дані, котрі притаманні саме до системи. Було зроблено величезну роботу, під час створення моків даних у базі, для того, щоб був датасет для навчання моделі. Так, булі використано 90% даних для навчання, а ще 10% залишилсь для того, щоб мануально протестувати рекомендації. Було виписано очікуємі рекомендації та перевірено, чи будуть вони отримані.

Так було помічено наступну відповідність, деякі ваги мали переважаюче значення, тому навіть маленький показник часу у комбінації з лайком на конкретному курсі міг однозначно додати курс, наприклад, алгоритмів до курсу з пайтону. Загалом, це дуже близькі речі за своїм призначенням, бо пайтон є доволі розповсюдженим, його легкість у вивчені надала йому широкий спектр використання. Так, його використовують у тому числі й науковці у своїй роботі, а ці самі науковці зазвичай й використовують чи структури даних, чи алгоритми, чи їх обох у комбінації. Тому цей приклад можна назвати вдалим.

Хоч можливості нейронної мережі для надання рекомендацій й були і є обмеженими, лімітованими у системі за визначенням. Варто нагадати, що є ліміт з позиції датасету, у якому було зазначено, що кількість даних є доволі обмеженою, бо потрібна реальна статистика у цьому домені, для того, щоб на основі неї можна було проводити реально навчання моделі.

Ще у ході тестування, було виявлено баг, що якщо дати кілька однакових векторів, то модель їх завчає, й при наданні будь-якого з них у якості тестових даних – було отримано вектор курсів, що напряму корелює з цим вхідним вектором для навчання. Так, це не дуже баг за своєю суттю, бо було помічено, що система добре регагує на скормлюємі їй дані, але це зайвий раз підтверджує, що треба бути пильним, під час аналізу даних ті їх очистки, чому було приділено особливу увагу, й додатково додано ще один слой до нейронної мережі, котрий відповідає за сплюскування вхідних векторів до іншої розмірності, так було двомірний вектор перетворено на одномірний, що наведено у прикладі у коді трохи вище, там, де

було сказано про деталі реалізації саме модулю нейронної мережі для надання рекомендацій.

Так, підводячи статистику по наданим рекомендаціям, варто помітити, що загальна тенденція є дуже приємною, можна сказати, що близько 65% рекомендацій, які було отримано відповідали очікуванням, до того ж, якщо ускладнювати алгоритми машинного навчання, обмежені обчислювальними потужностями, то можна досягти десь 70-75% точності, а якщо буде датасет з великою кількістю даних, то можна довести точність до всіх 85%. Нажаль, щоб перейти відмітку у 85%, портібно витратити багато часу працючи з великою професійною командою розробників та вчених, так, ресурсів направлених на цію роботу не є достатньо за кількісним фактором - це вже буде справжній виклик, що забиратиме занадто багато часу, як для однієї людини, для одного розробника – потрібна буде команда.

ВИСНОВКИ

У результаті роботи все описані цілі щодо методів аналізу рекомендаційних систем та вирішення задачі оптимізації часу за допомогою самих рекомендаційних систем шляхом влучних рекомендацій та можливості отримувати рекомендації онлайн, вивчаючи матеріл під час проходженн курсів. Так, були проаналізовані наступні методи рекомендування, й побудови персоналізованих рекомендацій:

- фільтраційний метод (конкретна, чи загальна);
- метод евристичного моделювання;
- поведінковий метод;
- метод колаборативної фільтрації.

Підсумовуючи, кожен з цих методів, варто зазначити ключові позитивні й негативні моменти, що призвели до розробки рекомендаційного модулю саме на основі колаборативної фільтрації, а не за використання будь-якого іншого методу персоналізації.

Так, фільтраційний метод є занадто простим, персоналізація за допомогою нього вимагає від користувача занадто багато інтеракцій з системою, тож такий метод не є ефективним, його використання лише незначно зможе зекономити час, під час пошуку курсів.

Далі, еврестичний метод – він є вже більш цікавим, бо людський розум машина ще не здатна обійти, так, можна було б й мануально давати рекомендації, чи власноруч створити групи «близьких» курсів, але тут є велика проблема у масштабувані. Наприклад те, що людина з плином часу перестане помічати частину інформації, а у голові тримати вже десяток близьки курсів стає складно, а якщо їх сотня і кожен з них має логічний зв’язок з іншим – тут вже не буде досить однієї людини, потрібно наймати кілька працівників, але вони можуть по-різному оцінювати близькість. Тому такий метод може мати місце лише у внутрішніх статичних системах, але для відкритих систем він є малоефективним.

Наступний метод – поведінковий, він вже є більш інтелектуальним методом. Можна було б, як каже цей метод, спостережити за поведінкою користувача й на

основі цього по поведінкових таблицях обирати той чи інший підхід, але все ж, він обмежений у зрості й вимагатиме постійного оновленн таких таблиць. Хоча загальна ідея підходу вже значно покращить показники попередніх методів, але це б вже був гібриндий метод, про який буде згадано трохи пізніше.

Наостанок, залишився обраний метод – найбільш інтелектуальний й цікавий. Це метод колаборативної фільтрації. Так, за допомогою створення профілю користувача, тут й є невелика подібність до поведінкого методу, бо для створення профілю потрібно записувати дії користувача. Але цей профіль вже буде не статично порівнюватись з іншими, а буде будуватись з різних фіч, таких як лайк, перегляд, час проведений вивчаючи, вектори, що у своїй комбінації й дають розуміння інтересів кожного конкретного користувача, так створюється саме профіль.

А для покращення результатів роботи, можна группувати всі підходи, так, кожний елемент системи можна будувати базуючись на конкретному методі персоналізації, але за загальну ідею все ж було обрано колаборативну філтьрацію, а знання з інших методів були виористані як плацдарм.

Так, було проведено аналіз предметної області й виявлено існуючі аналогічні системи з топу популярності та й з локальних ринків. Цікавим було дослідження алгоритмі, що використовуються у аналогах, та знайти варіанти оптимізацій. Так, було помічено, що всі аналоги, наукові праці, вся доменна область рекомендаційних систем еволюціонує. Розумовий труд постійно продовжується, а це каже, що всі люди прагнуть до розвитку, шляхом невпинного навчання й роботи над собою. Саме тому робота й є актуальною.

Попри всі недоліки, що здебільшого зумовлені обмеженнями й відсутністю фінансування, у цієї системи є великий потенціал, розроблений підхід можна застосувати за умови більшої кількості даних для тренування моделі, котрі можна ітеративно здобувати протягом життєвого циклу всієї системи, чи окремих її компонент.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Н.В. Голян, В.В. Голян, Л.Д. Самофалов. Алгебра понятий как формальный аппарат моделирования действий интеллекта над понятиями // Системи управління, навігації та зв`язку . – 2016. – Вип. 3(39). – С. 38 – 41.

2. Vira Golyan, Nataliia Golyan. EFFECTIVE METHODS OF INTELLIGENT ANALYSIS IN BUSINESS PROCESSES// 2019 IEEE . International Conference/ April. 2019.

3. Голян В.В., Самойленко Д.І. Создание архитектуры программной системы мониторинга информации про состаяние здоровья человека // Збірник наукових праць ХНУ ПС №1(59)2019р. – 6 с.

4. An Easy Introduction to Machine Learning Recommender Systems/ URL: https://www.kdnuggets.com/2019/09/machine-learning-recommender-systems.html (дата звернення: 12.09.2021)

5. Coursehunter / URL: https://coursehunter.net (дата звернення: 1.10.2021)

6. Coursera / URL: https://coursera.org (дата звернення: 1.10.2021)

7. Stepik / URL: https://stepik.org (дата звернення: 1.10.2021)

8. Udemy / URL: https://www.udemy.com (дата звернення: 1.10.2021)

9. Metanit / URL: https://metanit.com (дата звернення: 1.10.2021)

10. Steck, H., Baltrunas, L., Elahi, E., Liang, D., J. Deep Learning for Recommender Systems: A Netflix Case Study, 2021. – 701c.

11. Теорема Байеса: просто о сложном / URL: https://habr.com/ru/post/598979/ (дата звернення: 21.10.2021)

12. Rob F. Deep Statistical Learning, 2016. – 426p.

13. Тестирование производительности приложений и систем / URL: https://testmatick.com/ru/nagruzochnoe-testirovanie-testirovanie-proizvoditelnosti-prilozhenij-i-sistem/ (дата звернення: 23.12.2021)

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ ЗА НАУКОВИМИ НАПРЯМАМИ КЕРІВНИКА ТА НАУКОВЦІВ КАФЕДРИ ПРОГРАМНОЇ ІНЖЕНЕРІЇ

1. Н.В. Голян, В.В. Голян, Л.Д. Самофалов. Алгебра понятий как формальный аппарат моделирования действий интеллекта над понятиями // Системи управління, навігації та зв`язку . – 2016. – Вип. 3(39). – С. 38 – 41.

2. Vira Golyan, Nataliia Golyan. EFFECTIVE METHODS OF INTELLIGENT ANALYSIS IN BUSINESS PROCESSES// 2019 IEEE . International Conference/ April. 2019.

3. Голян В.В., Самойленко Д.І. Создание архитектуры программной системы мониторинга информации про состаяние здоровья человека // Збірник наукових праць ХНУ ПС №1(59)2019р. – 6 с.