МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДОНЕЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ВАСИЛЯ СТУСА

ПЕРЕПЕЛИЦЯ АРТЬОМ СЕРГІЙОВИЧ

Допускається до захисту:
в. о. завідувача кафедри
інформаційних технологій,
к. т. н, доцент
_____О. В. Зелінська
«____»_____ 2024 р.

ІНФОРМАЦІЙНА СИСТЕМА ДЛЯ УЗАГАЛЬНЕННЯ РЕЗУЛЬТАТІВ НАВЧАННЯ З ФУНКЦІЄЮ ПРОФОРІЄНТАЦІЇ

Спеціальність 122 «Комп’ютерні науки»

Кваліфікаційна (магістерська) робота

Науковий керівник:
Р. М. Бабаков, професор кафедри
інформаційних технологій,
д-р техн. наук, доцент
_____
(підпис)

Оцінка: ___/___/_____
(бали за шкалою ЄКТС/за національною шкалою)

Голова ЕК: _____
(підпис)

Вінниця 2024

Анотація

Перепелиця А. С. Інформаційна система для узагальнення результатів навчання з функцією профорієнтації. Спеціальність 122 “Комп’ютерні науки”, Освітня програма “Комп’ютерні технології обробки даних”. Донецький національний університет імені Василя Стуса, Вінниця, 2024.

У кваліфікаційній роботі досліджено існуючі рішення у галузі систем для узагальнення результатів навчання, наявність актуальних рішень щодо базової профорієнтації учнів. Розглянуто загальний підхід до розробки та архітектуру систем для узагальнення результатів навчання. Після чого розроблено таку систему та інтегровано в неї систему базової профорієнтації на основі штучного інтелекту.

Ключові слова: інформаційна система, профорієнтація, навчальний процес, психологічні тести, штучний інтелект, програмне забезпечення.

Abstract

Perepelytsia A. S. Information system for summarizing learning outcomes with career guidance function. Specialty 122 “Computer Science”, Educational program “Computer Data Processing Technologies”. Vasyl’ Stus Donetsk National University, Vinnytsia, 2024.

The qualification work investigated existing solutions in the field of systems for generalizing learning outcomes, the availability of relevant solutions for basic career guidance of students. The general approach to the development and architecture of systems for generalizing learning outcomes was considered. After that, such a system was developed and a basic career guidance system based on artificial intelligence was integrated into it.

Keywords: information system, career guidance, educational process, psychological tests, artificial intelligence, software.

ЗМІСТ

ВСТУП

Після завершення школи, старшокласники часто стикаються з важливим життєвим вибором: обрати майбутнє напрямок освіти та кар'єри. Проте, навіть з великою мотивацією, багато з них зіткнуться з невизначеністю. Необхідність вибору університету або професійної траєкторії може стати джерелом стресу та невпевненості, особливо коли відсутнє чітке уявлення про майбутні цілі та інтереси. Ця невизначеність часто призводить до витрати дорогоцінного часу на невдалий вибір [1] або навіть до ризику відмови від подальшої освіти. Не тільки це, але також велика ймовірність втрати можливості отримати безкоштовну освіту через необдуманість або непродумані рішення. Такі вагомі ризики створюють потребу в системному підході до підготовки старшокласників до вибору освітнього та кар'єрного шляху, спрямованого на розвиток їхніх здібностей, інтересів та усвідомлення можливостей на майбутнє.

Об'єктом дослідження є вчасне та влучне обирання студентами майбутнього шляху розвитку в залежності від їхніх індивідуальних здібностей, інтересів, а також результатів психологічних тестів та оцінок. Це явище має велике значення як для науки, так і для суспільства в цілому. Науковий інтерес полягає у розумінні того, як врахування цих факторів може сприяти успішній адаптації студентів на ринку праці та реалізації їхнього потенціалу.

Для суспільства ця проблематика стає актуальною в контексті ефективного використання людських ресурсів [2,3] та підтримки індивідуального розвитку кожної особистості. Влучний вибір майбутньої кар'єрної траєкторії може сприяти зниженню безробіття, розвитку економіки та соціальної стабільності. Тому дослідження цього процесу важливо для формування ефективної освітньої політики та створення системи професійної орієнтації, спрямованої на індивідуальні потреби кожного учня.

Предметом дослідження є інформаційна система для узагальнення результатів навчання з функцією професійної орієнтації старшокласників. Цей конкретний аспект об’єкта дослідження спрямовує увагу дослідника на розробку та впровадження інформаційної платформи, яка допоможе збирати, аналізувати та оцінювати дані про успішність та інтереси учнів з метою надання їм цільової професійної поради та підтримки в процесі вибору майбутнього освітнього та кар'єрного шляху.

Метою дослідження є розробка і впровадження інформаційної системи журналювання результатів навчання з функцією базової професійної орієнтації старшокласників з метою підтримки їхнього успішного вибору майбутнього освітнього та кар'єрного шляху.

Ця система буде спрямована на забезпечення індивідуального підходу до кожного учня, надання об'єктивних рекомендацій щодо найбільш відповідних напрямків навчання та професій, а також на підтримку їхнього самовизначення та розвитку. Метою є створення цілісного та функціонального інструменту, який сприятиме не лише влучному вибору кар'єрного шляху, а й розвитку особистості кожного старшокласника, забезпечуючи їм необхідні знання та підтримку на шляху до досягнення їхніх мет.

Завдання дослідження:

Розробити алгоритм збору та аналізу даних про успішність учнів, їхні інтереси та потенціал.

Провести аналіз психометричних тестів та оцінок, що визначають індивідуальні здібності та схильності старшокласників.

Розробити інформаційну систему, яка включатиме в себе інструменти журналювання та обробки даних.

Провести впровадження та тестування інформаційної системи на практиці з метою оцінки її ефективності.

Здійснити аналіз результатів впровадження та визначити можливі шляхи оптимізації та вдосконалення системи.

Підготувати звіт із результатами дослідження та рекомендаціями щодо подальших дій.

РОЗДІЛ 1
ТЕОРЕТИЧНІ ВІДОМОСТІ

1.1 Аналіз предметної області

Інформаційні системи журналювання результатів навчання стали невід'ємною частиною сучасної освітньої екосистеми, виконуючи ключову роль у покращенні управління навчальним процесом та підвищенні якості освіти. Ці системи автоматизують рутинні завдання, знижуючи навантаження на вчителів та адміністративний персонал, що дозволяє їм більше часу приділяти безпосередньо навчальному процесу та роботі зі студентами.

Однією з головних переваг інформаційних систем журналювання результатів навчання є централізоване зберігання даних, що дозволяє зручно зберігати й обробляти інформацію про студентів, їхню успішність та прогрес. Це забезпечує легкий доступ до необхідної інформації для всіх учасників освітнього процесу: вчителів, студентів, батьків та адміністраторів. Сучасні інформаційні системи журналювання результатів навчання також мають інтерактивні функції, такі як електронні щоденники, платформи для спільної роботи, форуми та чати, що сприяє покращенню комунікації та підтримці колективного навчання[4].

Аналітика та звітність є ще однією важливою складовою інформаційних систем журналювання результатів навчання[5]. Ці системи дозволяють збирати та аналізувати великі обсяги даних про навчальний процес, виявляти тенденції, прогнозувати результати та приймати обґрунтовані рішення щодо покращення якості освіти. Завдяки цьому освітні заклади можуть своєчасно виявляти слабкі місця в навчальному процесі та вживати необхідних заходів для їх усунення, що сприяє досягненню кращих результатів.

Інформаційні системи журналювання результатів навчання значно підвищують ефективність управління навчальним процесом, автоматизуючи рутинні завдання та забезпечуючи зручний доступ до інформації. Це підвищує прозорість навчального процесу та сприяє активній участі батьків у

освітньому процесі. Студенти та їхні батьки отримують зручний доступ до інформації про успішність, розклад занять, домашні завдання та інші важливі дані.

Однак, впровадження інформаційних систем журналювання результатів навчання пов'язане і з певними викликами. Технічні аспекти, такі як необхідність значних ресурсів та інфраструктури, можуть бути складними для освітніх закладів з обмеженими фінансовими можливостями. Безпека та конфіденційність даних є ще одним важливим питанням, адже зберігання та обробка великої кількості персональних даних студентів вимагає високого рівня захисту. Крім того, впровадження нових технологій може зустрічати опір з боку користувачів, які звикли до традиційних методів ведення документації. Успішне впровадження інформаційних систем журналювання результатів навчання потребує належного навчання та підтримки користувачів.

Сучасні інформаційні системи журналювання результатів навчання все частіше інтегруються з іншими освітніми технологіями, такими як системи управління навчанням (LMS), платформи для онлайн-тестування та інструменти для адаптивного навчання. Це забезпечує більш комплексний підхід до управління навчальним процесом та покращує взаємодію між різними компонентами освітньої екосистеми.

Мобільні додатки стали ще однією важливою тенденцією у розвитку інформаційних систем журналювання результатів навчання. З розвитком мобільних технологій, все більше систем надають можливість доступу до своїх функцій через мобільні додатки, що забезпечує зручність та гнучкість для користувачів.

Таким чином, інформаційні системи журналювання результатів навчання вже є важливим інструментом для сучасної освіти, проте існує ще одна потенційно цінна функція, яка могла б розширити можливості цих систем і допомогти студентам у визначенні свого майбутнього шляху. Це функція

базової профорієнтації, яка враховувала б як академічні результати, так і результати проходження психологічних тестів[6].

Впровадження профорієнтаційної функції в інформаційні системи журналювання результатів навчання дозволило б використовувати накопичені дані про академічну успішність студентів та результати психологічних тестів для створення персоналізованих рекомендацій щодо вибору професії.

Використовуючи оцінки з різних предметів, інформацію про позашкільну активність, інтереси, навички студентів та результати психологічних тестів, система могла б автоматично генерувати рекомендації щодо потенційних професійних шляхів, які найкраще відповідають їхнім сильним сторонам, інтересам та психологічним особливостям.

Це могло б мати кілька важливих переваг:

Підвищення усвідомленості: Студенти, особливо старшокласники, часто стикаються з труднощами у виборі майбутньої професії. Персоналізовані рекомендації на основі їхніх академічних результатів та психологічних тестів можуть допомогти їм краще зрозуміти свої сильні сторони та потенційні сфери діяльності.

Зменшення стресу та невизначеності: Вибір професії — це серйозний крок, який часто супроводжується стресом та сумнівами. Автоматизовані рекомендації, засновані на комплексному аналізі академічних досягнень та психологічних тестів, можуть знизити цей стрес, надаючи студентам обґрунтовані підказки, які базуються на їхніх реальних досягненнях, інтересах та психологічних особливостях.

Покращення підготовки до ринку праці: Використовуючи дані про успішність студентів та їхні психологічні профілі, система може також враховувати поточні тенденції на ринку праці, попит на певні професії та необхідні навички. Це дозволить студентам орієнтуватися на ті професійні напрями, які будуть затребувані в майбутньому.

Індивідуальний підхід: Завдяки персоналізованому підходу до профорієнтації, система може пропонувати студентам варіанти додаткових

курсів або позашкільних заходів, які допоможуть розвинути необхідні навички та підготуватися до обраної професії. Психологічні тести, зокрема, можуть виявити особливості характеру та темпераменту, які є важливими для певних професій.

Впровадження профорієнтаційної функції в інформаційні системи журналювання результатів навчання потребуватиме розвитку аналітичних інструментів та алгоритмів, які зможуть ефективно обробляти великі обсяги даних і генерувати точні рекомендації. Використання штучного інтелекту та машинного навчання може стати ключовим аспектом цього процесу, дозволяючи створити адаптивні системи, які враховуватимуть індивідуальні особливості кожного студента.

Загалом, додавання функції базової профорієнтації до інформаційних систем журналювання результатів навчання могло б суттєво покращити якість освітнього процесу, забезпечуючи студентам додаткову підтримку у виборі їхнього майбутнього шляху. Це не лише підвищило б ефективність навчання, але й сприяло б формуванню більш впевнених та підготовлених до професійного життя випускників.

1.2 Огляд аналогів

Було оглянуто найбільш відомі в Україні електронні журнали, далі надається їх порівняльна характеристика:

1. NZ[7] (Нове знання)

Опис:

NZ (Нове знання) - український науковий журнал, присвячений різним галузям науки, включаючи гуманітарні та природничі дисципліни. Журнал публікує наукові статті, дослідження, рецензії та огляди.

Особливості:

Тематика: Широкий спектр наукових дисциплін.

Мови публікацій: Українська та англійська.

Аудиторія: Вчені, дослідники, викладачі, студенти.

Формат: Електронні публікації, доступні онлайн безкоштовно.

Мета: Сприяння розвитку науки та обміну знаннями.

2. **E-Journal IEA**[8] (Електронний журнал Інституту освітньої аналітики)

Опис:

E-Journal від Інституту освітньої аналітики - це платформа для ведення електронних журналів та щоденників у закладах загальної середньої освіти. Вона була створена для спрощення документообігу та забезпечення доступу до освітньої інформації.

Особливості:

Цільова аудиторія: Учні, батьки, вчителі, адміністрація шкіл.

Функціонал: Ведення оцінок, відвідуваності, розкладу занять, збирання освітньої статистики.

Безпека: Захищеність даних, можливість інтеграції з іншими системами.

Безкоштовність: Державна безкоштовна система для всіх закладів освіти України.

Мета: Підвищення ефективності управління освітою та доступність освітньої інформації для всіх учасників процесу.

3. **Всеосвіта**[9]

Опис:

Всеосвіта також пропонує електронний журнал для шкіл, що дозволяє вчителям вести облік успішності учнів, розклад занять та інші освітні документи в електронному вигляді.

Особливості:

Цільова аудиторія: Вчителі, учні, адміністрація шкіл.

Функціонал: Ведення оцінок, облік відвідуваності, розклад уроків.

Доступність: Платформа доступна для використання в школах по всій Україні.

Мета: Полегшення роботи вчителів та забезпечення прозорості у навчальному процесі.

4. **Atoms**[10]

Опис:

Atoms - онлайн платформа для ведення електронних журналів та щоденників у закладах освіти. Платформа інтегрована з національними освітніми системами та забезпечує високий рівень безпеки даних. Особливості:

Цільова аудиторія: Вчителі, учні, адміністрація шкіл, батьки.

Функціонал: Ведення оцінок, домашні завдання, облік відвідуваності, розклад занять.

Інтеграція: Платформа інтегрована з Автоматизованим інформаційним комплексом освітнього менеджменту (АІКОМ).

Безпека: Комплексна система захисту інформації, затверджена Державною службою спеціального зв'язку та захисту інформації України.

Мета: Забезпечення ефективного управління освітнім процесом та доступу до освітніх даних для всіх учасників процесу

Висновок

Кожен з цих онлайн журналів має свої унікальні особливості та цілі. Якісь з них більш універсальні, інші ж більш спрямовані на певні цілі, закладені їх розробниками, проте в жодному з них не реалізовано функціоналу, котрий я пропоную.

1.3 Постановка задачі

На основі виконаного аналізу предметної області можна сформулювати постановку задачі.

Наша задача полягає в тому, щоб створити кросплатформений додаток для журналювання результатів навчання, імплементувати в нього функціонал проходження тестів чи отримання відповідних результатів через інтеграцію з

вже існуючими сервісами проходження тестування. Результати накопичених даних про навчання та результати проходження тестів мають оброблятися додатком та на виході давати один чи кілька рекомендаційних результатів щодо того які спеціальності можуть зацікавити певного учня.

Основні функціональні можливості:

- Доступ до журналу з різних пристроїв (комп'ютери, планшети, смартфони) та браузерів
- Інтуїтивно зрозумілий та зручний для використання інтерфейс для вчителів, учнів та батьків
- Розподілення відображення таблиць з оцінками учнів по місяцям для більш зручної навігації по журналу та економії трафіку на передачі даних за певний період а не в цілому
- Можливість введення, перегляду та редагування оцінок
- Створення ієрархії користувачів типу Власник – Адміністратор – Викладач – Учень
- Можливість генерування детальних звітів про академічний прогрес
- Можливість генерувати звіти в кінці навчального року
- Можливість отримати профорієнтаційну оцінку
- Адаптивність інтерфейсу під більшість актуальних конфігурацій
- Створення інтуїтивно зрозумілого інтерфейсу для батьків та учнів
- Унеможливити зміну оцінок викладачами більш ніж за певний період в минуле, задля підвищення безпеки
- Посилання щомісячних виписок батькам на електронну пошту з результатами навчання їхньої дитини

Основні нефункціональні можливості:

- Безпека даних: Захист особистих даних учнів та викладачів від несанкціонованого доступу, використання шифрування та інших заходів безпеки.

- Масштабованість: Можливість обслуговування великої кількості користувачів та даних без зниження продуктивності системи.
- Резервне копіювання та відновлення даних: Регулярне резервне копіювання даних та можливість їх відновлення у разі втрати або пошкодження.
- Гнучкість в налаштуваннях оцінювання: Можливість налаштовувати різні типи оцінювання відповідно до потреб конкретного навчального закладу та предмету.
- Підтримка стандартів: Відповідність шкільним нормативам та стандартам оцінювання.
- Історія змін: Збереження історії змін оцінок для відстеження та вирішення будь-яких суперечностей або питань.

Можливі способи розвитку системи:

- Інтеграція з навчальними сервісами тестування як-то «НА УРОК», «justclass» тощо.
- Інтеграція з сервісами онлайн дзвінків як-то Google Meet, Zoom тощо.
- Створення функціонала онлайн щоденника
- Створення мобільних додатків на IOS та Android
- Створення додаткових систем мотивації всередині системи як-то турнірна дошка, командне суперництво тощо.

Висновок до розділу 1

У розділі обговорено важливість впровадження інформаційних систем, які дозволяють автоматизувати процеси управління навчальним процесом і забезпечують можливість інтеграції функції профорієнтації. Було визначено, що сучасні інформаційні системи для ведення журналів навчальних досягнень значно спрощують адміністративні процеси та дозволяють освітнім закладам зосереджуватися на підвищенні якості навчання. Такі системи сприяють ефективному збору, аналізу та обробці інформації про академічні досягнення учнів, забезпечуючи прозорість і доступність даних як для учнів, так і для

батьків та вчителів. Проте, незважаючи на значний прогрес у цій сфері, жодна з існуючих систем повністю не охоплює функцію профорієнтації, яка могла б допомагати старшокласникам влучно обирати майбутній освітній та кар'єрний шлях на основі їх академічних досягнень і психологічних тестів.

Було розглянуто кілька популярних інформаційних систем (наприклад, NZ, E-Journal IEA, Всеосвіта та Atoms), які використовуються в українських освітніх закладах. Незважаючи на їх переваги, жодна з них не реалізує профорієнтаційні функції. Впровадження подібного інструменту може значно полегшити процес вибору професії для учнів, зменшивши невизначеність та стрес, пов'язані з прийняттям такого важливого рішення. Рекомендації, побудовані на основі аналітики навчальних результатів і психологічних тестів, сприятимуть успішній адаптації молоді до професійного життя.

Таким чином, розвиток таких систем, як профорієнтаційна платформа, що інтегрується з освітніми журналами, є важливим кроком у сучасній освіті. Це дозволить старшокласникам отримувати індивідуальні поради щодо вибору кар'єри на основі аналізу їхніх успіхів і особистих схильностей, що позитивно вплине на їхнє майбутнє навчання і професійний розвиток..

РОЗДІЛ 2
ТЕОРЕТИЧНА ЧАСТИНА РОЗРОБКИ

2.1 Вибір архітектури системи

Наша задача полягає в створенні кросплатформеного клієнт-серверного додатка. Аналізуючи сучасний ринок технологій, було вирішено обрати клієнт-серверну архітектуру для створення даної системи через наступні її переваги:

Кросплатформеність: Клієнт-серверна архітектура дозволяє розробникам створювати серверну частину, яка може працювати на будь-якій операційній системі, і клієнтські додатки, які можуть бути написані для різних платформ (Windows, macOS, Linux, iOS, Android тощо). Це забезпечує доступність і зручність використання програмного забезпечення на різних пристроях та операційних системах, що значно розширює аудиторію користувачів.

Гнучкість: Клієнт-серверна модель забезпечує високу гнучкість в управлінні ресурсами та розширенні системи. Завдяки модульності, можна легко оновлювати, додавати нові функції або змінювати існуючі компоненти без необхідності повного перероблення системи. Серверна частина може обслуговувати різні типи клієнтів (мобільні додатки, веб додатки, десктопні програми), що дозволяє легко інтегрувати нові технології та пристрої.

Сучасність: Клієнт-серверна архітектура відповідає сучасним вимогам до розподілених систем та забезпечення високої продуктивності. Завдяки можливості масштабування, можна забезпечити обслуговування великої кількості клієнтів і високий рівень продуктивності навіть при значному навантаженні. Сучасні серверні технології, такі як мікросервіси, хмарні обчислення та контейнеризація, ще більше підвищують ефективність і надійність клієнт-серверних рішень.

Централізоване управління даними: Клієнт-серверна архітектура дозволяє централізовано зберігати та обробляти дані на сервері, що спрощує управління даними, забезпечує їх безпеку та дозволяє ефективно використовувати ресурси. Централізоване управління також полегшує резервне копіювання, відновлення даних та контроль доступу до інформації.

Поліпшена безпека: У клієнт-серверній архітектурі сервер відповідає за аутентифікацію користувачів і управління доступом, що дозволяє забезпечити вищий рівень безпеки. Конфіденційні дані зберігаються на сервері, а не на клієнтських пристроях, що знижує ризики витоку інформації при втраті або компрометації клієнтського пристрою.

2.2 Особливості будови бекенд частини

2.2.1 Вибір мови програмування та фреймворку

Для бекенд частини були взяті за основу мова програмування Kotlin[11] та фреймворк Spring Boot[12].

Spring Boot відомий своєю швидкістю ініціалізації проекту та конвенційним підходом до конфігурації, що дозволяє розробникам ефективно розгортати додатки з мінімальними зусиллями. Також цей фреймворк має велику к-ть модулів(Рис 2.1), що дозволяють швидко та ефективно розробляти додатки різного рівня складності.

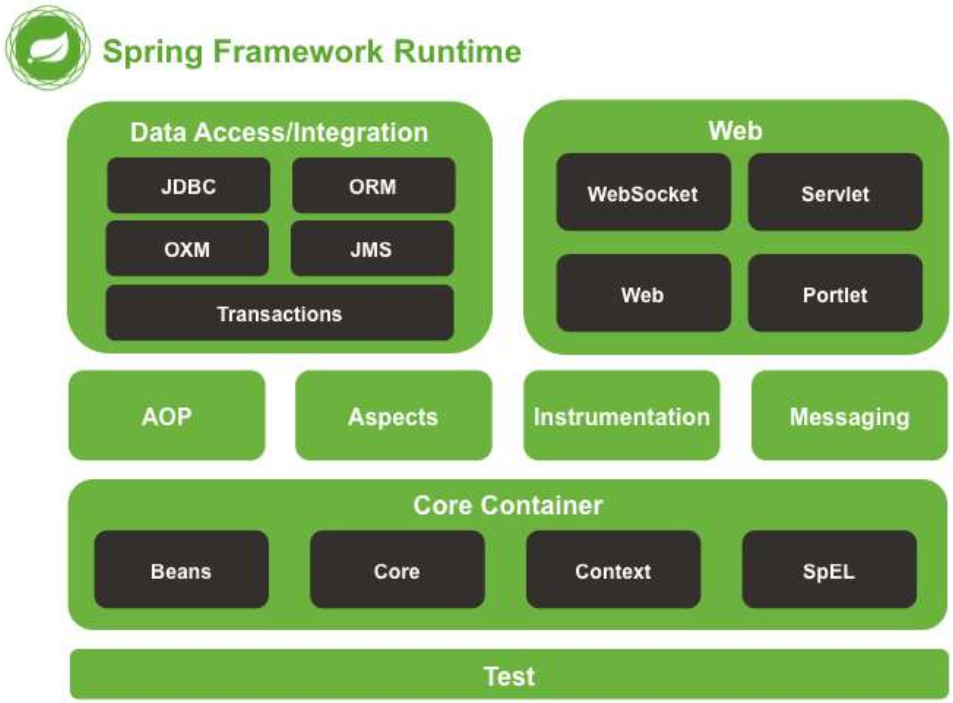


Рисунок 2.1 Структура модулів фреймворку Spring

Хоча цей фреймворк загально відомий більше за його орієнтованості на мову програмування Java, проте мова програмування Kotlin, завдяки тому, що також запускається на Java Virtual Machine, є інтероперабельною з нею, тобто код Kotlin може викликати Java-код без обмежень та проблем. Крім того, код Kotlin є більш безпечним та лаконічним [13], що зменшує кількість потенційних помилок, що можуть здійснювати розробники під час написання коду.

Також варто зазначити, що інтеграція Kotlin зі Spring Boot дозволяє значно підвищити продуктивність розробників за рахунок спрощеного синтаксису та кращої підтримки функціональних програмних конструкцій. Завдяки своїй лаконічності, код на Kotlin скорочується в середньому на 40% у порівнянні з аналогічним Java-кодом, що знижує складність підтримки та модифікації додатків. Крім того, Kotlin забезпечує повну сумісність з існуючими бібліотеками та фреймворками Java, що дозволяє легко використовувати напрацьовану інфраструктуру та коду.

Використовуючи мову програмування Kotlin з фреймворком SpringBoot ми отримуємо наступні переваги:

- **Типова безпека (Null-Safety).** Kotlin надає вбудовані механізми для захисту від помилок, пов'язаних із null-посиланнями, які є однією з найбільших проблем у Java. Це дозволяє зменшити кількість винятків NullPointerException, що робить код більш надійним. Також це більш зручний та керований спосіб керування Nullable-об’єктами, ніж прийнятий зараз у Java підхід через об’єкт Optional.
- **Функціональне програмування**. Kotlin підтримує функціональний стиль програмування, що дозволяє використовувати лямбда-вирази, вищі функції, а також іммутабельні структури даних. Це забезпечує більш декларативний підхід до написання коду, який краще відповідає сучасним принципам розробки.
- **Спрощена обробка даних**. Kotlin надає багаті можливості для роботи з колекціями через вбудовані функції для фільтрації, трансформації та агрегації даних, що дозволяє мінімізувати рутинний код.
- **Розширені функції**. Завдяки підтримці розширюваних функцій (extension functions), Kotlin дозволяє легко додавати нові методи до існуючих класів без необхідності змінювати їх код або створювати підкласи, що особливо корисно в контексті роботи з фреймворками.

2.2.2 Вибір СУБД для зберігання даних

Для зберігання даних додатку потрібно було обрати СУБД. Було проведено порівняльне дослідження можливостей та переваг кількох СУБД. За основу була взята СУБД PostgreSQL, з нею порівнювалися такі СУБД як MySQL, SQLite, MariaDB. Були отримані такі результати:

PostgreSQL [14-16]

Переваги:

Повна підтримка стандарту SQL: PostgreSQL має найповнішу відповідність стандарту SQL серед безкоштовних СУБД, забезпечуючи складні транзакції, агрегатні функції, підзапити та підтримку вбудованих функцій для роботи з JSON та XML.

Розширюваність: PostgreSQL підтримує збережені процедури на різних мовах (PL/pgSQL, PL/Python тощо), а також дозволяє створювати користувацькі функції та типи даних, що робить її надзвичайно гнучкою для розробки кастомних рішень.

Масштабованість: Система оптимізована для обробки великих обсягів даних та складних запитів, що робить її ідеальною для корпоративних систем і масштабованих програм.

Безпека та надійність: Підтримка ACID-транзакцій, механізмів контролю доступу та перевірка цілісності даних забезпечують високий рівень надійності та безпеки.

Підтримка паралелізму: PostgreSQL оптимізована для виконання паралельних запитів, що підвищує продуктивність системи в умовах високих навантажень.

Недоліки:

Складність налаштування: Складність налаштування для новачків може бути вищою порівняно з деякими іншими СУБД.

MySQL[17, 18]

Переваги:

Швидкість: MySQL є легкою та швидкою СУБД, добре підходить для додатків, які не потребують складної обробки даних або великої кількості транзакцій.

Простота: MySQL відомий своєю простотою налаштування, що робить її популярним вибором для простих веб-додатків і невеликих систем.

Широка підтримка: Ця СУБД має велику спільноту користувачів та безліч інтеграцій з популярними веб-фреймворками.

Недоліки:

Обмеженість транзакцій: MySQL не завжди забезпечує повну підтримку ACID-транзакцій, особливо у випадку з таблицями типу MyISAM.

Брак розширюваності: Можливості розширення та кастомізації MySQL обмежені порівняно з PostgreSQL, що може бути суттєвим недоліком для великих або спеціалізованих проєктів.

Низька підтримка паралельних запитів: MySQL поступається PostgreSQL у продуктивності при роботі з великими обсягами даних і складними запитами.

SQLite[19, 20]

Переваги:

Легкість та вбудованість: SQLite є "легкою" базою даних, яка не вимагає серверного середовища. Вона часто використовується в мобільних і вбудованих додатках.

Простота використання: Для початку роботи з SQLite не потрібно налаштовувати сервер, що робить її ідеальною для невеликих проєктів або для швидкої розробки прототипів.

Невеликий розмір: SQLite має дуже малий розмір виконуваного файлу, що робить її привабливою для програм з обмеженими ресурсами.

Недоліки:

Масштабованість: SQLite не підходить для великих або багатокористувацьких систем через відсутність підтримки одночасних запитів і обмежену продуктивність при роботі з великими обсягами даних.

Обмежена функціональність: На відміну від PostgreSQL, SQLite має обмежені можливості для складної обробки даних, а її підтримка розширюваних функцій та кастомних рішень обмежена.

MariaDB[21, 22]

Переваги:

Сумісність з MySQL: MariaDB є форком MySQL, що дозволяє легко мігрувати з MySQL на MariaDB без змін у коді або архітектурі.

Швидкість: MariaDB може запропонувати дещо кращу продуктивність у порівнянні з MySQL завдяки деяким оптимізаціям.

Відкрита ліцензія: На відміну від MySQL, MariaDB розвивається як повністю відкрите ПЗ без жодних корпоративних обмежень.

Недоліки:

Менш розвинена екосистема: Порівняно з PostgreSQL, MariaDB має менш розвинену підтримку кастомних рішень і обмежену масштабованість у складних проєктах.

Обмежена функціональність: Подібно до MySQL, MariaDB має обмежені можливості в роботі з великими реляційними системами та складними транзакціями.

На основі вище вказаних даних була створена порівняльна таблиця (Таб. 2.1)

Таблиця 2.1 – Порівняльна характеристика СУБД

База даних	**Підтримка ACID**	**Розширюваність**	**Продуктивність для великих систем**	**Підтримка паралельних запитів**	**Складність налаштування**
PostgreSQL	Повна	Широка	Висока	Висока	Вища складність
MySQL	Часткова	Обмежена	Середня	Низька	Низька
SQLite	Обмежена	Немає	Низька	Відсутня	Дуже проста
MariaDB	Часткова	Обмежена	Середня	Низька	Низька

Зважаючи на комплексність системи, що має бут розроблена, було обрано PostgreSQL.

2.3 Програмні засоби використовувані під час розробки

2.3.1 Середовище розробки

2.3.1.1 Бекенд

Для розробки бекенд частини було обрано середовище розробки під назвою IntelliJ IDEA.

IntelliJ IDEA[23] — це інтегроване середовище розробки (IDE), розроблене компанією JetBrains. Відоме своєю гнучкістю та багатофункціональністю, це IDE отримало визнання серед розробників завдяки своїй здатності підвищувати продуктивність та забезпечувати зручний робочий процес. Одна з головних переваг IntelliJ IDEA — це підтримка широкого спектра мов програмування, включаючи Java, Kotlin, Groovy, Scala, і навіть певною мірою вебтехнології, такі як HTML, CSS, і JavaScript.(Рис. 2.2)

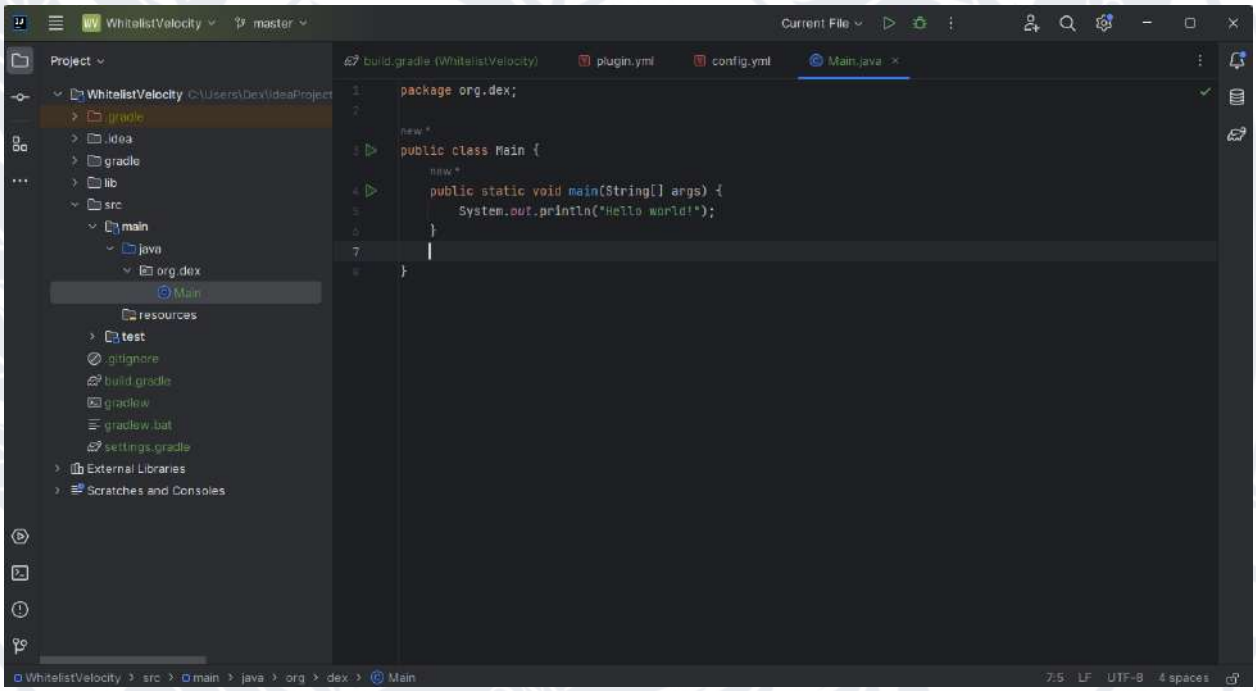


Рисунок 2.2 Користувацький інтерфейс IntelliJ IDEA

Переваги IntelliJ IDEA

Розумний редактор коду: IntelliJ IDEA не лише допомагає писати код, але й активно аналізує його під час написання. Завдяки вбудованій системі інтелектуальних підказок, це IDE надає рекомендації щодо поліпшення коду, пропонує варіанти автодоповнення та виявляє помилки в режимі реального часу.

Інтеграція з системами контролю версій: Для командної розробки чи маштабних проектів важливим є інтеграція з Git, Mercurial та іншими системами контролю версій. IntelliJ IDEA надає інтуїтивно зрозумілий інтерфейс для управління репозиторіями, гілками, коммітами та відкотами.

Рефакторинг та оптимізація: Функції рефакторингу в IntelliJ IDEA дозволяють безпечно змінювати структуру коду, не порушуючи його функціональність. Це особливо корисно для великих проектів, де важливо підтримувати чистоту коду та високу читабельність

Підтримка Docker та Kubernetes: Інтеграція з Docker і Kubernetes робить IntelliJ IDEA потужним інструментом для розгортання контейнеризованих додатків та управління середовищем розробки, що дозволяє автоматизувати процеси розробки та забезпечити їх стабільність.

Недоліки IntelliJ IDEA

Вимоги до системних ресурсів: IntelliJ IDEA є доволі вимогливим до ресурсів системи, особливо при роботі з великими проектами. Це може впливати на продуктивність на слабких комп'ютерах.

Платність: Хоча існує безкоштовна версія — Community Edition, більшість розширених функцій доступні лише в платній версії — Ultimate Edition.

2.3.1.2 Фронтенд

Для розробки фронтенду було обрано середовище розробки під назвою WebStorm(Рис. 2.3).

WebStorm[24] — це інтегроване середовище розробки (IDE), спеціально створене для веб-розробників компанією JetBrains. Завдяки своїм розширеним можливостям і підтримці сучасних вебтехнологій, WebStorm є одним із

найпопулярніших інструментів серед розробників JavaScript, TypeScript, HTML, CSS та інших фронтенд-технологій.

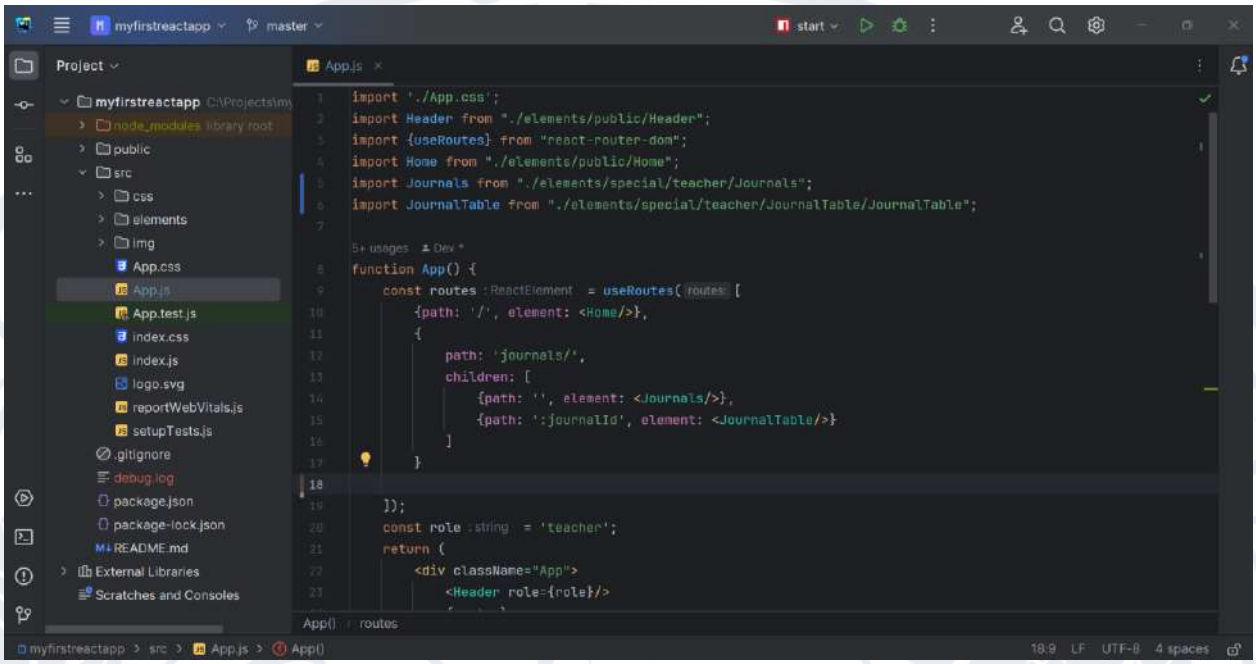


Рисунок 2.3 Користувацький інтерфейс WebStorm

Переваги WebStorm

Підтримка сучасних вебтехнологій: WebStorm підтримує всі основні фронтенд-технології, включаючи JavaScript, TypeScript, Angular, React, Vue.js, Node.js. IDE забезпечує автодоповнення, інспекції коду та підказки для всіх цих технологій, що значно прискорює роботу з ними.

Інтеграція з системами контролю версій: Як і інші IDE від JetBrains, WebStorm має потужну інтеграцію з Git, GitHub, Mercurial, що дозволяє легко керувати репозиторіями, вирішувати конфлікти та відслідковувати зміни у коді.

Налагодження та тестування: Вбудовані інструменти для налагодження (debugging) та тестування дозволяють знаходити та виправляти помилки безпосередньо в IDE. Підтримка Node.js та браузерного налагодження дає змогу працювати з різними середовищами прямо з WebStorm.

Інтеграція з популярними інструментами розробки: WebStorm підтримує інтеграцію з такими популярними інструментами розробки, як npm, Webpack, Babel та інші, що робить робочий процес ще більш гнучким та зручним для розробників.

Інспекції коду та рефакторинг: Як і в IntelliJ IDEA, WebStorm надає потужні інструменти для автоматичного виправлення помилок, рефакторингу та оптимізації коду, що допомагає підтримувати чистоту та читабельність проектів

2.3.2 Віртуалізація

Для зручності та уникнення можливих проблем під час встановлення, СУБД PostgreSQL була імплементована як контейнер у Docker-і.

Docker[25] — це платформа для контейнеризації, що дозволяє розробникам упаковувати додатки та їх залежності в автономні контейнери, які можуть запускатися на будь-якому сервері. Використання Docker вирішує багато проблем, пов'язаних із залежностями, сумісністю та середовищами розробки, дозволяючи розробникам створювати надійні та легко масштабовані додатки.

Переваги Docker

Контейнеризація додатків: Docker дозволяє створювати ізольовані контейнери для кожного додатку або його компонента. Це гарантує, що кожен контейнер має власне середовище та залежності, що мінімізує ризик конфліктів між додатками.

Портативність: Контейнери Docker можуть запускатися на будь-якому сервері, що підтримує Docker, незалежно від операційної системи. Це робить

додатки більш переносними між різними середовищами (розробка, тестування, продакшн).

Швидкість розгортання: Контейнери запускаються набагато швидше, ніж віртуальні машини, оскільки не вимагають завантаження гостьової операційної системи. Це дозволяє швидко розгортати та масштабувати додатки у великих системах.

Ефективність використання ресурсів: Docker використовує ресурси більш ефективно, ніж традиційні віртуальні машини, оскільки всі контейнери працюють на одній операційній системі, використовуючи її ядро.

Інтеграція з CI/CD: Docker легко інтегрується з інструментами безперервної інтеграції та доставки (CI/CD), такими як Jenkins, GitLab CI, CircleCI, що дозволяє автоматизувати тестування, складання та розгортання додатків.

Недоліки Docker

Складність налаштування мережі: Робота з мережею всередині контейнерів може бути складною для новачків, особливо при роботі з багатокомпонентними системами.

2.3.3 Системи збору проектів

2.3.3.1 Бекенд

Для збору бекенд частини було обрано систему збору проектів Gradle.

Gradle[26] — це система автоматизації збірки, яка стала невід'ємною частиною процесу розробки програмного забезпечення, особливо для проектів на базі мов Java, Kotlin та Groovy. Спроектований для забезпечення високої гнучкості та масштабованості, Gradle поєднує переваги декларативного підходу, що використовується в Maven, і програмного стилю конфігурації, як у Ant. Gradle відомий своєю здатністю автоматизувати безліч процесів, від

компіляції коду до розгортання та тестування, що робить його одним із найбільш популярних інструментів для великих і складних проектів.

Переваги Gradle

Висока продуктивність: Gradle оптимізований для роботи з великими проектами, пропонуючи механізм інкрементальної збірки, який значно скорочує час на повторні компіляції. Він відстежує зміни у файлах та виконує тільки ті завдання, які необхідні, що економить ресурси і час.

Підтримка багатомодульних проектів: Gradle особливо добре підходить для багатомодульних проектів, які складаються з кількох підсистем або компонентів. Можливість легкої організації модулів дозволяє працювати над різними частинами проекту незалежно один від одного та спрощує інтеграцію всіх компонентів.

Гнучкість конфігурації: Завдяки використанню мови конфігурації Groovy або Kotlin DSL, Gradle дозволяє розробникам створювати динамічні та гнучкі сценарії збірки. Це дає можливість додавати нові завдання, налаштовувати поведінку збірки та інтегрувати різні інструменти та фреймворки.

Інтеграція з екосистемами Gradle: підтримує інтеграцію з різними інструментами та системами, включаючи Maven, Ant, Jenkins, GitLab CI та інші. Це дозволяє безперешкодно використовувати Gradle у вже існуючих процесах розробки та безперервної інтеграції.

Підтримка популярних мов і платформ: Gradle підтримує багато мов програмування та платформ, включаючи Java, Kotlin, Groovy, Scala, Android, C++, і Swift. Це робить його універсальним інструментом для проектів будь-якої складності та технологічної основи.

Інтеграція з Docker та Kubernetes: Gradle також може бути використаний для створення Docker-образів, інтеграції з системами оркестрації, такими як Kubernetes, що дозволяє автоматизувати розгортання додатків у контейнерах.

Недоліки Gradle

Крива навчання: Для новачків Gradle може здаватися складним, особливо через програмну природу конфігурацій та відсутність простого XML-підходу, як у Maven.

Ресурсоємність: Хоча Gradle оптимізований для продуктивності, для дуже великих проектів його інкрементальні збірки можуть бути вимогливими до ресурсів, що вимагає налаштувань для оптимальної роботи.

2.3.3.2 Фронтенд

Для розробки фроентенд частини було обрано систему збору проектів Vite.

Vite[27] — це нова генерація інструментів для збірки та розробки JavaScript-додатків, що зосереджується на швидкості та ефективності. Створений спочатку для роботи з Vue.js, Vite також відмінно підходить для React та інших сучасних фронтенд-фреймворків. Він поєднує в собі можливості ES-модулів у браузері та потужність сучасних збірок для того, щоб забезпечити миттєвий старт розробки та швидке оновлення сторінок. Vite активно використовується в екосистемі React завдяки своїй здатності скоротити час компіляції та забезпечити швидший процес розробки.

Переваги Vite

Швидкий сервер розробки: Vite використовує інший підхід у порівнянні з традиційними інструментами, надаючи ES-модулі безпосередньо в браузер без попереднього збирання. Це дозволяє миттєво запускати сервер розробки, незалежно від розміру проєкту, на відміну від традиційних бандлерів, які можуть потребувати часу на обробку.

Гарячий модульний перезавантажувач (HMR): HMR у Vite оптимізований для швидкості, і оновлення застосовуються менш ніж за 50 мілісекунд у більшості випадків. Це особливо корисно при розробці з React, оскільки зміни можна бачити одразу без необхідності перезавантажувати сторінку.

Потужне збирання на базі Rollup: Для продакшн-збирання Vite використовує Rollup, який забезпечує ефективне розділення коду та оптимізацію розмірів бандлу. Це робить Vite ідеальним як для односторінкових додатків (SPA), так і для великих вебсайтів.

Простота конфігурації: Vite потребує мінімум налаштувань. Ви можете почати роботу з React лише за кілька рядків коду, тоді як для Webpack або Parcel зазвичай потрібно більше часу та налаштувань. Vite також автоматично виявляє залежності та обробляє лише те, що необхідно.

Сучасні технології: Vite спочатку підтримує сучасні можливості JavaScript, такі як ECMAScript-модулі (ESM), тоді як багато старіших інструментів, таких як Webpack, стикаються з труднощами адаптації до нового стандарту.

Швидке зростання популярності

Спільнота навколо Vite швидко розширюється, і це відображає сучасний попит на швидші та простіші інструменти для збирання проєктів.

Недоліки Vite

Інтеграція з більш старими браузерами: Підтримка старих браузерів може потребувати додаткової конфігурації, оскільки Vite орієнтований на сучасні стандарти. Для додатків, які мають працювати на старих версіях браузерів, необхідно додати поліфіли або інші механізми підтримки.

Екосистема плагінів: Хоча Vite активно розвивається і підтримує безліч плагінів, його екосистема поки не така широка, як у Webpack. Це може створювати певні труднощі при інтеграції специфічних плагінів або рішень, які вже мають більш усталену підтримку в інших інструментах.

2.4 Особливості будови фронтенд частини

Для створення фронтенд-частини продукту було обрано мову програмування JavaScript з використанням фреймворку React[28].

JavaScript [29] є мовою програмування, яка дозволяє реалізувати на веб-сторінках динамічну взаємодію та інтерактивність, перетворюючи статичні сторінки в інтерактивні інтерфейси.

Переваги JavaScript

Широка підтримка: Завдяки підтримці всіх основних веб-браузерів, JavaScript забезпечує безперебійну роботу веб-додатків на різних платформах. Це означає, що розробники можуть бути впевнені, що їхній код буде працювати на більшості пристроїв.

Велика екосистема: JavaScript має велику кількість бібліотек та фреймворків, таких як React, Angular, і Vue.js, що спрощують створення динамічних і інтерактивних користувацьких інтерфейсів. Це дозволяє зменшити час розробки і підвищити ефективність.

Асинхронність: JavaScript підтримує асинхронні операції, що робить його ідеальним для роботи з даними, отриманими з серверів. Використання

таких механізмів, як Promise та async/await, спрощує управління асинхронними запитами.

Простота в навчанні: JavaScript є відносно легкою мовою для початківців. З чіткою синтаксичною структурою і великою кількістю ресурсів для навчання, нові розробники можуть швидко освоїти основи.

Гнучкість: JavaScript дозволяє розробникам писати код у різних стилях програмування, включаючи об'єктно-орієнтоване, функціональне та імперативне програмування. Це робить мову підходящою для різних проектів.

ReactJS - це бібліотека, розроблений компанією Facebook, що створена для створення користувацьких інтерфейсів, вона використовує компонентний підхід та віртуальний DOM для ефективного оновлення інтерфейсу.

Використання JavaScript разом з ReactJS надає нам наступні переваги:

Компонентна архітектура: ReactJS використовує компонентну архітектуру, що дозволяє розробникам розбивати інтерфейс на незалежні, повторно використовувані компоненти. Це спрощує розробку, тестування та підтримку коду, оскільки кожен компонент можна розробляти та перевіряти окремо.

Віртуальний DOM: Однією з ключових переваг ReactJS є використання віртуального DOM, що забезпечує високу продуктивність додатку. Завдяки віртуальному DOM зміни відстежуються ефективніше, що дозволяє мінімізувати операції з реальним DOM і підвищити швидкість рендерингу.

Односторонній потік даних: ReactJS підтримує односторонній потік даних, що робить управління станом додатку простішим і передбачуванішим. Це знижує ймовірність помилок і полегшує відстеження змін у стані додатку, що є важливим аспектом для складних інформаційних систем.

Розширення можливостей JavaScript: ReactJS значно розширює можливості JavaScript, додаючи потужні інструменти для створення інтерактивних інтерфейсів. Завдяки цьому розробники можуть створювати

сучасні, динамічні веб додатки, які відповідають найвищим стандартам зручності та продуктивності.

2.5 Бібліотеки для створення інтерфейсу користувача

Для створення користувацького інтерфейсу будемо використовувати бібліотеки Bootstrap та MaterialUI.

Bootstrap[30]**:** є одним з найпопулярніших фреймворків для розробки інтерфейсів користувача. Ми будемо використовувати останню, п’яту версію, адже вона пропонує суттєві поліпшення порівняно з попередніми версіями, зокрема відмову від залежності від jQuery, що робить її легшою і швидшою.

Серед особливостей цієї бібліотеки хотілося б відзначити наступні:

Гнучка сітка: Bootstrap 5 використовує сучасну 12-колонну сіткову систему, яка дозволяє легко створювати адаптивні макети.

Компоненти: Бібліотека містить безліч готових компонентів, таких як кнопки, форми, навігаційні панелі, модальні вікна тощо.

Утиліти: Включає набір утиліт для швидкого стилізування елементів без написання додаткового CSS.

Темізація: Простий механізм для створення тем, що дозволяє легко змінювати зовнішній вигляд сайту.

Документація: Вичерпна та зрозуміла документація.

MaterialUI (MUI)[31] є реалізацією концепції Material Design від Google для React. Ця бібліотека надає компоненти, що відповідають гайдлайнам Material Design, що робить її ідеальною для створення сучасних та естетичних інтерфейсів.

Ключовими особливостями цієї бібліотеки є:

Компоненти: Широкий набір компонентів, включаючи кнопки, карти, діалоги, меню та інші, що відповідають стандартам Material Design.

Сумісність з React: Глибока інтеграція з React, що робить MUI відмінним вибором для проектів, побудованих на цій бібліотеці.

Стилізація: Потужні можливості для стилізації компонентів за допомогою системи стилів MUI та CSS-in-JS.

Адаптивність: Всі компоненти автоматично адаптуються до різних розмірів екранів.

Естетика: Дизайн, що відповідає сучасним тенденціям та рекомендаціям Google.

Модульність: Компоненти можуть використовуватися незалежно один від одного.

2.6 Вибір бібліотек для обробки даних

Для обробки даних відповідно до наших вимог було розглянуто такі бібліотеки, як Apache Mahout, Apache Spark MLlib, H2O.ai, Elasticsearch, Seldon. Ось невеличкий опис кожної з них:

Apache Mahout[32] — це бібліотека машинного навчання, яка спеціалізується на алгоритмах колаборативної фільтрації, кластеризації та класифікації. Вона побудована на екосистемі Hadoop, що забезпечує високу масштабованість та здатність обробляти великі обсяги даних. Mahout добре підходить для побудови рекомендаційних систем завдяки своїм ефективним алгоритмам для обробки взаємодій користувачів та предметів.

Основною складністю при роботі з Apache Mahout є необхідність налаштування Hadoop-кластера, що може бути викликом для новачків. Втім, для великих проектів з обробкою великих даних Mahout є надійним вибором.

Apache Spark MLlib[33] — це бібліотека машинного навчання, інтегрована з Apache Spark, що дозволяє ефективно виконувати обчислення на розподілених кластерах. MLlib підтримує широкий спектр алгоритмів, включаючи колаборативну фільтрацію, кластеризацію та регресію. Ця бібліотека підходить для задач, що вимагають високої продуктивності та масштабованості.

Spark MLlib підтримує Java, Scala, Python та R, що робить її універсальною для різних команд розробників. Вона має хорошу документацію та активну спільноту. Основним викликом є необхідність знання Spark для ефективного використання MLlib, що може вимагати додаткового навчання для новачків.

H2O.ai[34, 35] — це потужна бібліотека машинного навчання з відкритим вихідним кодом, яка підтримує широкий спектр алгоритмів, включаючи градієнтний бустинг, нейронні мережі, випадкові ліси та класифікацію. H2O.ai відзначається своєю здатністю працювати з різними типами даних та високою масштабованістю завдяки розподіленим обчисленням.

H2O.ai має зручний інтерфейс, підтримуючи Java, Python, R та веб-інтерфейс Flow, що спрощує процес розробки та тестування моделей. Основними плюсами є її зрозуміла документація та простота у використанні навіть для новачків у сфері машинного навчання.

Elasticsearch — це розподілена пошукова та аналітична система, яка може бути використана для побудови рекомендаційних систем завдяки своїм потужним можливостям пошуку та кластеризації. Основна сила Elasticsearch полягає у векторному пошуку, що дозволяє ефективно знаходити схожі об'єкти.

Elasticsearch надає простий у використанні REST API та підтримує різні клієнти для інтеграції з іншими системами. Висока масштабованість та можливість обробки великих обсягів даних роблять його привабливим вибором для проектів, що вимагають потужного пошуку та аналізу даних. Однак, Elasticsearch краще підходить для текстових даних і може бути менш гнучким для інших типів завдань машинного навчання.

Seldon — це платформа для розгортання та управління моделями машинного навчання за допомогою контейнерів та мікросервісів. Вона дозволяє легко інтегрувати моделі, створені в різних бібліотеках машинного навчання, та забезпечує високу масштабованість і надійність.

Seldon підтримує широкий спектр алгоритмів, включаючи градієнтний бустинг, нейронні мережі та випадкові ліси, і надає зручний REST API для інтеграції. Основною перевагою є можливість швидкого розгортання моделей у виробничих середовищах та управління ними. Seldon підходить для команд,

які шукають гнучке рішення для розгортання та масштабування своїх моделей машинного навчання.

Проаналізувавши ці бібліотеки, відповідно до нашого завдання, було обрано бібліотеку H2O.ai.

Це обумовлено наступними причинами:

Гнучкість та різноманітність моделей: H2O.ai підтримує багато алгоритмів, що дозволяє вибрати найкращий підхід для конкретного завдання.

Простота використання: Простий API та веб-інтерфейс роблять цю бібліотеку зручною для швидкого прототипування та експериментування.

Підтримка різних типів даних: Легко обробляє числові та категорійні дані, що важливо для обробки оцінок та результатів психологічних тестів.

2.7 Вибір психологічних тестів для формування цілісної картини щодо поведінки та схильностей учня

Для старшокласників вибір професії часто є складним зокрема через недостатню обізнаність про власні здібності та інтереси. Саме тому все більше уваги приділяється використанню комплексних психологічних тестів, які допомагають краще зрозуміти особисті схильності, мотивації та професійні нахили учнів. Завдяки цим інструментам можна виявити сильні сторони індивідуальності, рівень мотивації до досягнень, структуру інтелекту та інші ключові параметри, що дозволяють сформувати обґрунтовані рекомендації щодо майбутньої професійної діяльності. Важливість такого підходу полягає у сприянні успішній адаптації молоді на ринку праці та розкритті їхнього потенціалу у сфері, яка їм найбільше відповідає.

Під час дослідження наявних тестів, що б надавали подібну інформацію було розглянуто:

1. Тест Голланда (RIASEC)[36]

Цей тест є одним із найвідоміших інструментів для виявлення професійних нахилів та інтересів. Він ґрунтується на теорії Джона Голланда, яка стверджує, що задоволення від професії залежить від збігу типу

особистості людини з характеристиками роботи (Рис. 2.4)[16]. Голланд виділяє шість типів особистості, і кожен з них відповідає певним професійним сферам. Тест допомагає з'ясувати, які види діяльності найбільше підходять людині та які професії можуть забезпечити найбільше задоволення.

Відповідність типів особистості типам професійного середовища

Типи особистості	Типи професійного середовища					
	Р	І	С	К	П	А
Реалістичний	++	+	-	+	-	- -
Інтелектуальний	+	++	-	-	- -	+
Соціальний	- -	-	++	-	+	+
Конвенційний	+	-	+	++	+	- -
Підприємницький	-	- -	+	-	++	+
Артистичний	-	-	+	- -	-	+ +

Рисунок 2.4 Відповідність типів особистості типам професійного середовища[36]

2. Професійна діагностика за Клімовим (ДДО)[37]

Ця методика широко використовується для профорієнтації учнів та студентів. Клімов запропонував класифікацію професій за типами взаємодії з об’єктами праці, що дозволяє зрозуміти, які типи діяльності будуть найцікавішими та найпродуктивнішими для конкретної людини. Тест допомагає студентам вибрати професію, яка відповідає їхнім інтересам і здібностям, ґрунтуючись на тому, як вони взаємодіють із людьми, технікою, природою чи інформацією.

3. Тест Айзенка на визначення типу темпераменту(EPQ)[38]

Цей тест оцінює чотири основні типи темпераменту – сангвінік, холерик, флегматик і меланхолік. Темперамент впливає на поведінкові особливості людини, її реакцію на стрес, соціальні взаємодії та загальний спосіб життя.

Тест Айзенка дозволяє дізнатися, як емоційні та психологічні особливості можуть впливати на професійний вибір і успіх у різних видах діяльності.

4. Тест структури інтелекту Амтхауера (IST)

Тест Амтхауера[39, 40] оцінює інтелектуальні здібності в кількох напрямках: вербальний інтелект, математичні та логічні здібності, просторове мислення. Це комплексна методика, яка дозволяє визначити сильні сторони учня в різних інтелектуальних сферах, що є важливим при виборі інтелектуально складних професій. Тест підходить для тих, хто розглядає технічні чи наукові професії.

5. Тест Кеттелла (16PF)[41, 42]

Цей тест складається з 16 факторів особистості і дає детальний профіль індивідуальних психологічних характеристик людини. Тест допомагає зрозуміти, як риси особистості впливають на поведінку в різних ситуаціях, у тому числі в професійній сфері. Тест Кеттелла часто використовується для профорієнтації, оскільки він враховує як соціальні, так і емоційні аспекти особистості, що важливо для вибору підходящої професії.

6. Методика Люшера (Колірний тест)

Тест Люшера[43, 44] дозволяє визначити психоемоційний стан людини на основі її переваг до кольорів. Він оцінює рівень стресу, емоційну напруженість та наявність внутрішніх конфліктів. Методика допомагає визначити, наскільки старшокласник готовий до професій, що вимагають високого рівня емоційної стійкості та концентрації, а також підходить для розуміння емоційних схильностей у професійній діяльності.

7. Методика Т. Елерса (Тест на мотивацію досягнень)[37, 45]

Цей тест оцінює рівень мотивації до досягнень та схильність до уникання невдач. Він допомагає зрозуміти, чи старшокласник прагне до успіху або боїться поразок, а також який рівень цілеспрямованості він демонструє. Це важливий показник для вибору професії, оскільки мотивація до досягнень впливає на успішність у навчанні та професійній діяльності.

8. Карта інтересів Голомштока[46, 47]

Ця методика призначена для виявлення професійних інтересів старшокласників у різних сферах діяльності, таких як наука, техніка, мистецтво або комунікація. Тест допомагає визначити основні напрямки, що цікавлять учня, і на основі цього підібрати відповідні професії.

Таблиця 2.4 – Порівняльна характеристика розглянутих психологічних тестів.

Тест	**Основний фокус**	**Особливості учня, що досліджуються**	**Мета**	**Приблизний час проходження**
Тест Голланда (RIASEC)	Професійні нахили	Інтереси до певних видів діяльності, тип особистості (реалістичний, артистичний тощо)	Визначення відповідних професій на основі інтересів	30-40 хвилин
Тест Айзенка на темперамент	Тип темпераменту	Емоційність, стресостійкість, рівень активності та соціальної взаємодії	Вибір професії відповідно до типу темпераменту	10-15 хвилин
Професійна діагностика за Клімовим (ДДО)	Типи професій	Ступінь взаємодії з об’єктами праці (людина-людина, людина-техніка тощо)	Рекомендації щодо професій за типом діяльності	20-30 хвилин
Тест Кеттелла (16PF)	Особистісні характеристики	Емоційна стабільність, домінантність, товариськість, сумлінність (16 факторів особистості)	Детальна характеристика особистості для вибору професії	45-60 хвилин

Продовження таблиці 2.4

Тест Люшера (Колірний тест)	Психоемоційний стан	Емоційний стан, рівень стресу, внутрішні конфлікти	Визначення емоційного стану для професійної орієнтації	5-10 хвилин
Тест на мотивацію досягнень Елерса	Мотивація досягнень	Ступінь мотивації досягати успіху, уникати невдач, рівень цілеспрямованості	Оцінка прагнення до успіху в професійній діяльності	15-20 хвилин
Карта інтересів Голомштока	Професійні інтереси	Схильність до різних сфер діяльності (наука, техніка, мистецтво, комунікація)	Визначення основних інтересів для профорієнтації	20-25 хвилин

В результаті можемо розділити тести умовно на 2 категорії (тести для визначення професійних нахилів та тести для визначення складу особистості) та обрати найбільш ефективні тести:

Найкращі тести для визначення складу особистості:

Тест Кеттелла (16PF) – комплексний аналіз особистісних характеристик.

Тест Айзенка – визначає темперамент і психологічні особливості.

Шкала інтернальності-екстернальності Роттера – виявляє рівень відповідальності й самоконтролю.

Найкращі тести для визначення професійних нахилів:

Тест Голланда (RIASEC) – підходить для узгодження професійних інтересів із особистісними характеристиками.

Професійна діагностика за Клімовим – допомагає обрати сферу професійної діяльності.

Тест професійних уподобань Голланда – детальніший аналіз професійних нахилів.

2.8 Механізм перевірки токенів доступу та обробка помилок авторизації

Під час використання вебдодатку важливо щоб користувацький досвід був гарним, тому було розроблено систему (Рисунок 2.4), при якій при запиті даних з ресурсного серверу перевірялись-би права доступу та, за необхідності, оновлювалися токени доступу та відновлення. Токени доступу використовуються для забезпечення безпеки та контролю доступу до захищених ресурсів. Описаний механізм перевіряє валідність токена, його термін дії та відповідність прав доступу. У разі успішної перевірки користувач отримує доступ до ресурсу, якщо токен доступу буде невалідний а токен відновлення валідний, то система автоматично оновить токени і надішле їх назад користувачу. У випадках коли токени або відсутні або обидва не є валідними - система повертає помилку 403 ("Заборонено"), що вказує на відсутність дозволу на виконання запиту

Висновок до розділу 2

У другому розділі було проведено детальний аналіз вибору архітектури та технологій для розробки інформаційної системи. Основним вибором стала клієнт-серверна архітектура, яка забезпечує гнучкість, масштабованість та зручність управління даними. Було обрано мову програмування Kotlin та фреймворк Spring Boot[48] для бекенд частини, а також СУБД PostgreSQL для зберігання даних через її розширюваність і надійність.

Для фронтенд частини використано JavaScript разом із React, що забезпечує інтерактивність користувацького інтерфейсу. Було детально

розглянуто вибір середовищ розробки та інструментів для збору проєктів, таких як Gradle для бекенду та Vite для фронтенду, що сприяють ефективності розробки.

Таким чином, розділ надає чітке розуміння архітектури системи та використаних технологій, підкреслюючи важливість сучасних інструментів для забезпечення продуктивності, безпеки та гнучкості в роботі інформаційної системи.

РОЗДІЛ 3 ПРАКТИЧНА ЧАСТИНА

3.1 Створення плану розробки

Перед розробкою додатку важливо структурувати вимоги до додатку, проаналізувати його функціональні можливості та способи взаємодії користувачів з додатком. Це дозволить нам створити певний план розробки та мати уявлення щодо прогресу під час всього періоду створення продукту.

Одним з головних інструментів дослідження можливих способів взаємодії користувача з додатком в наш час є так звані сценарії використання.

Сценарії використання — це описи конкретних ситуацій, у яких користувач взаємодіє з додатком для досягнення певної мети. Вони відображають послідовність дій, які виконує користувач, та реакції системи на ці дії. Такі сценарії допомагають виявити основні функціональні вимоги, перевірити зручність інтерфейсу, а також знайти потенційні слабкі місця у взаємодії. Використання сценаріїв дозволяє краще зрозуміти потреби цільової аудиторії, визначити основні пріоритети функціоналу та сприяти більш ефективному процесу розробки. Вони також є корисним інструментом для тестування, адже дозволяють створити реалістичні сценарії перевірки роботи системи.

В нашому додатку всіх користувачів можна розділити на категорії по їх ролям у системі. Розпишемо основні необхідні можливості щодо взаємодії користувачів певної ролі з додатком:

Студент: Користувачі з цією роллю будуть найбільш численними у нашому додатку. До їх основних функцій будуть входити наступні сценарії використання:

- Авторизація у додатку
- Перегляд власних оцінок

- Проходження психологічних тестів
- Отримання рекомендацій щодо профорієнтації

Викладач: Другий по численності клас користувачів. Для них передбачимо наступні сценарії використання:

- Авторизація
- Створення нових уроків у журналі
- Виставлення оцінок за певними видами роботи на уроці
- Перегляд всіх оцінок у журналі

Адміністратор: Малочисельний клас користувачів в межах додатку. Мають більше важливих можливостей а відповідно і сценаріїв використання додатку:

- Авторизація
- Реєстрація нових користувачів
- Коригування оцінок у журналах
- Створення нових журналів
- Визначення списку учнів для журналу
- Визначення списку вчителів, що б мали доступ до журналу
- Деактивація аккаунтів

Власник: Зазвичай це має бути лише один користувач з найбільшими можливостями керування в додатку. Його можливості будуть ширшими за адміністраторів, і в результаті отримуємо наступні попередні сценарії використання:

- Авторизація
- Реєстрація нових користувачів
- Коригування оцінок у журналах
- Створення нових журналів
- Визначення списку учнів для журналу

- Визначення списку вчителів, що б мали доступ до журналу
- Деактивація аккаунтів
- Зміна користувачам їх ролей
- Перегляд логів системи для контролю за роботою

3.2 Розробка дизайну додатку

Для проекту «Інформаційна система для узагальнення результатів навчання з функцією профорієнтації» був обраний мінімалістичний дизайн, він має бути достатньо простим щоб пересічний користувач зміг легко розібратися в ньому та користування додатком не викликало в нього ніяких труднощів. Основні сторінки для ведення журналу додатку будуть створені у вигляді таблиць(Рис. 3.1) по аналогії з звичайними, фізичними, журналами, проте будуть і ключові відмінності.

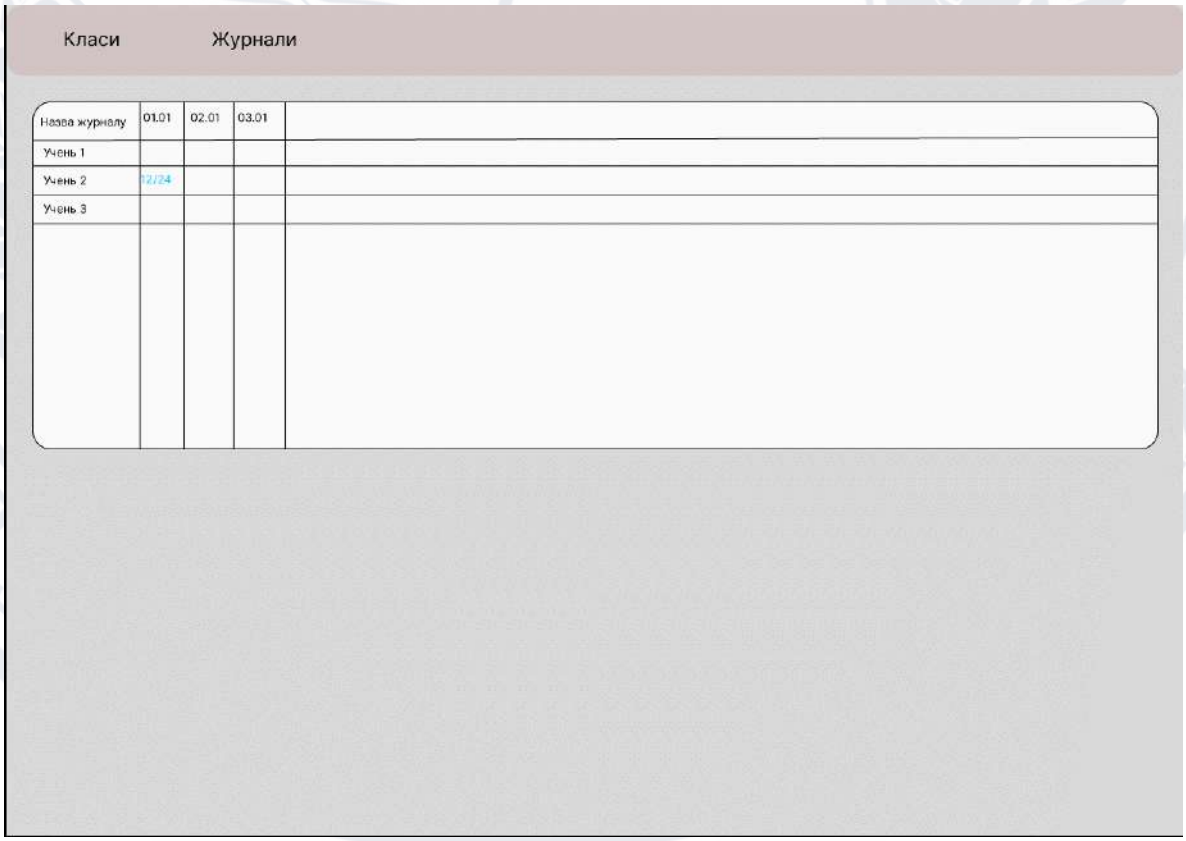


Рисунок 3.1 Базовий макет сторінки ведення журналу

Зокрема, на відміну від фізичних журналів, де часто виникали складнощі з тим щоб в межах одного уроку відмітити роботу студента кількома оцінками за різні види роботи, в нашому додатку кожна клітинка буде показувати суму балів що студент заробив на занятті та максимальну к-ть балів що він міг заробити, це уникне ситуації коли кілька низьких оцінок у журналі будуть створювати ілюзію що учень гарно відповів один раз(рис. 3.2).

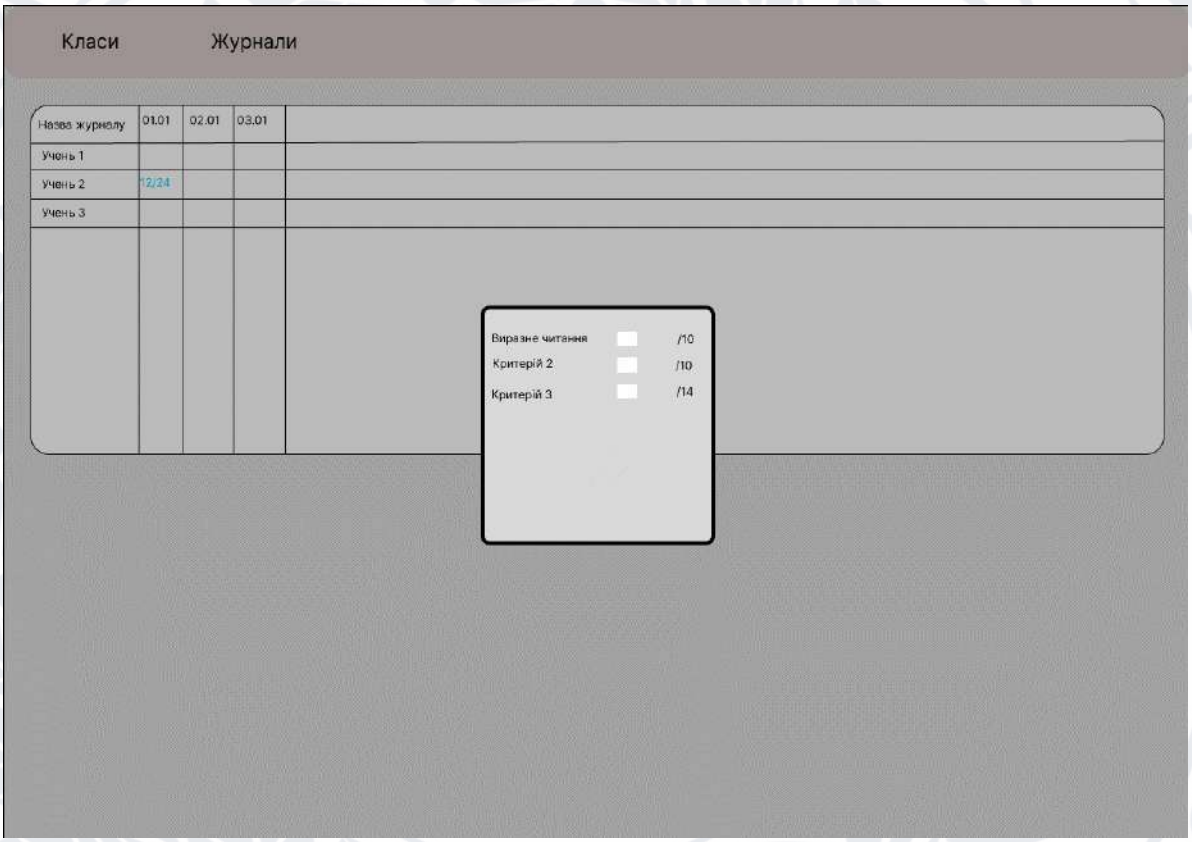


Рисунок 3.2 Прототип вікна виставлення оцінок

Макет навігаційної панелі має бути лаконічним, мати лише необхідні кнопки для навігації по додатку.

Для більш зручного відображення елементів керування, на невеликих екранах та екранах мобільних телефонів, ми будемо приховувати елементи керування у окреме меню, що за вимогою буде виїжджати з-за краю екрану.

На сторінці вибору журналу всі журнали будуть сортуватися за предметами, до яких відносяться ці журнали. Це спростить розуміння

приналежності журналів до предметів без необхідності створювати громіздкі назви для журналів.

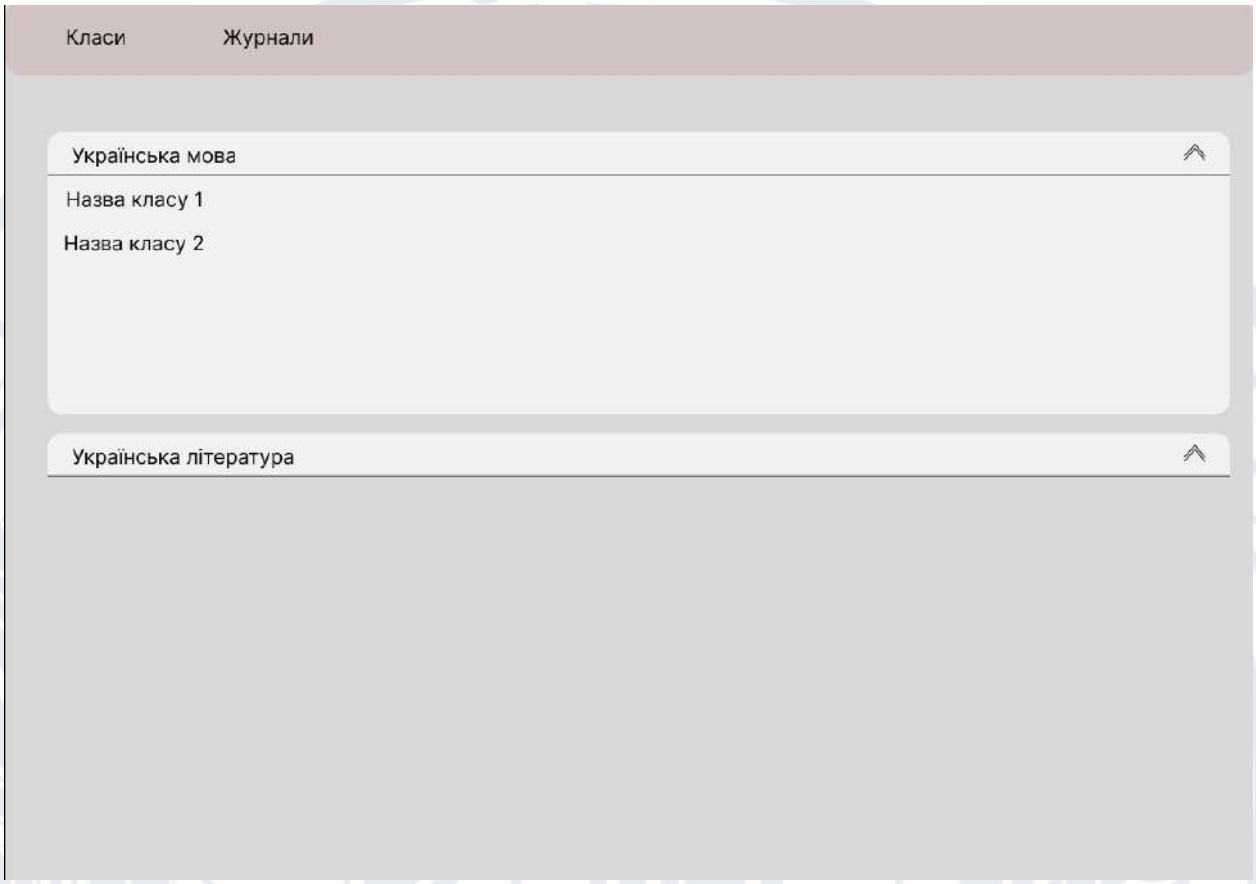


Рисунок 3.3 Прототип сторінки вибору журналу

Прототипування дизайну було виконано на такому ресурсі як Figma[49, 50].

Figma — це сучасний інструмент для створення інтерфейсів, який працює повністю у веб-браузері, а також має десктопні версії. Її головна перевага для нашого проекту – можливість створювати функціональні прототипи дизайну, це дає можливість обдумати певні дизайнерські рішення ще на ранніх етапах розробки проекту.

До функціональних можливостей фігми, для створення інтерактивних дизайнів, можна віднести:

- **Прототипування**: Figma дозволяє створювати інтерактивні прототипи прямо з дизайнерських макетів. Ви можете додати

переходи між екранами, анімації, клікабельні кнопки, імітуючи роботу реального додатку чи сайту.

- **Інтерактивні компоненти**: Використовуючи інтерактивні компоненти, можна створювати елементи з різними станами (наприклад, кнопка "Hover", "Disabled"). Це значно спрощує тестування функціональності та презентацію готового дизайну.
- **Тестування прототипів**: Дизайн можна поділитися через посилання, і користувачі або клієнти зможуть взаємодіяти з прототипом, як з реальним додатком, без потреби додаткового ПЗ.
- **Спільна робота**: Команда розробників може отримувати доступ до готових макетів, а завдяки інтегрованим коментарям легко узгоджувати зміни або доповнення.
- **Плагіни та інтеграції**: Figma підтримує численні плагіни, які допомагають автоматизувати робочі процеси: від перевірки доступності до експорту готових макетів у код.

3.3 Розробка функціоналу додатку

3.3.1 Авторизація.

Авторизація у додатку працює на основі JWT токенів доступу, що зберігається у HTTPOnly Cookie.

JWT[51] — це компактний стандартний формат для передачі інформації між сторонами у вигляді JSON-об'єктів, які закодовані та підписані, щоб забезпечити їх достовірність. Ключовими перевагами JWT є те, що він дозволяє безпечно зберігати інформацію про користувача чи сеанс, зменшуючи потребу у частому звертанні до бази даних. JWT складається з трьох частин: заголовка (header), корисного навантаження (payload) і підпису (signature). Після генерації токену сервер підписує його секретним ключем або парою приватного/публічного ключів, що дозволяє легко перевіряти його

автентичність. Завдяки цьому підхід з JWT є швидким, масштабованим і добре підходить для розподілених систем.

HTTPOnly Cookie[52] — це спеціальний тип cookie-файлів, які доступні лише на рівні сервера і не можуть бути використані JavaScript у браузері. Це значно знижує ризики атак XSS (Cross-Site Scripting), оскільки шкідливий скрипт не матиме доступу до токену, що зберігається у cookie. HTTPOnly Cookie забезпечують додатковий рівень безпеки для конфіденційної інформації, такої як токени доступу, що робить їх ідеальними для роботи з JWT. Крім того, ці cookie автоматично передаються клієнтом у запитах до сервера, що спрощує управління сесією і підвищує зручність для користувача.

Зберігання JWT у **HTTPOnly Cookie** поєднує переваги обох технологій, забезпечуючи високий рівень безпеки і зручності. Завдяки тому, що токен зберігається у HTTPOnly Cookie, він залишається невидимим для JavaScript, що захищає його від XSS-атак. Крім того, cookie автоматично додається до HTTP-запитів, що усуває необхідність вручну керувати токеном на стороні клієнта, як це відбувається при зберіганні токену в LocalStorage або SessionStorage. Це не тільки підвищує безпеку, але й спрощує реалізацію механізму авторизації, оскільки браузер самостійно обробляє передачу токену. Таким чином, поєднання JWT та HTTPOnly Cookie дозволяє створити просту, безпечну і масштабовану систему авторизації.

Зберігаються у HTTPOnly Cookie власне токен доступу та токен відновлення. Токен доступу надає доступ користувачу до ресурсів, проте час його життя досить короткий. І коли він стає більше не актуальним, в діло вступає токен відновлення. Він в собі зберігає специфічні дані що допомагають в межах того самого запиту відновити токен доступу та оновити токен відновлення, що забезпечує безперервне користування додатком користувачу без необхідності повторно авторизуватися.

3.3.2 Зберігання оцінок

Оцінки зберігаються у відповідних таблицях у реляційній СУБД PostgreSQL. Для забезпечення достатньої гнучкості при масштабуванні додатку були відокремлені такі сутності:

Journal: Власне журнал, ціль цього класу – керувати доступом до журналу.

Lesson: Зберігає в собі дані про заняття як-то тема заняття та домашнє завдання.

Mark: Зберігає в собі дані необхідні для ідентифікації на якому уроці кому та яку саме оцінку виставив викладач.

WorkType: За задумкою додаток був в першу чергу налаштований на 100-бальну систему що останнім часом почали впроваджувати у школах, тому для забезпечення необхідної гнучкості при створенні наперед продуманих видів робіт та їх максимальних балів оцінювання, був створений такий клас що б кожен викладач міг створювати унікальні для свого предмету види роботи.

3.3.3 Рекомендації щодо профорієнтації

Для створення рекомендацій щодо профорієнтації для першої ітерації були створені синтетичні дані щодо того які шкільні предмети більш важливі для яких галузей, на основі цього за допомогою потужного інструменту H2O.AI створена модель ШІ що в подальшому була експортована як файл у форматі MOJO.

MOJO (Model Object, Optimized) — це формат збереження та розгортання моделей машинного навчання, розроблений компанією H2O.ai. Він дозволяє експортувати навчену модель з середовища H2O для подальшого використання в будь-якому іншому додатку або середовищі. MOJO забезпечує

швидку і ефективну інтеграцію моделей завдяки легковаговій структурі, яка включає всі необхідні метадані та залежності для виконання прогнозів.

Однією з ключових переваг MOJO є можливість використовувати моделі без потреби в окремому сервері H2O, що робить цей формат ідеальним для розгортання в реальному часі, навіть в умовах обмежених ресурсів. Крім того, формат MOJO підтримує моделі, створені різними алгоритмами H2O. Завдяки високій продуктивності та простоті інтеграції, MOJO широко використовується для розробки production-ready рішень у галузях, що потребують швидкого прогнозування, таких як фінанси, охорона здоров'я, маркетинг та промислові додатки.

Далі цей файл було імпортовано в робоче середовище через бібліотеку h2o-core, реалізовано спеціальний PredictionService та створено методи для передбачення топ-5 найбільш відповідних галузей знань що можуть зацікавити учня.

3.3.4 Структура проекту

Проект поділений на різні рівні такі як:

- **Рівень домену**: Тут зберігаються всі класи що зберігають дані, від основних до тих, що створені тільки щоб передати дані в певному форматі між елементами додатку.
- **Рівень репозиторіїв**: Тут зберігаються класи та інтерфейси що відповідають за взаємодію зовнішніми сервісами як-то СУБД, наприклад за збереження оцінки учня, створення нового уроку, тощо.
- **Рівень сервісів**: Тут зберігаються класи що реалізують бізнес-логіку в додатку, тобто через них реалізований весь основний функціонал додатку

- **Рівень конфігурації**: Тут зберігаються всі класи що відповідають за глобальну конфігурацію додатку
- **Рівень безпеки**: Тут зберігаються всі класи що захищають ресурси додатку від несанкціонованого доступу, займаються перевіркою та створенням токенів доступу, тощо.

3.3.5 Система оцінювання знань

Для оцінювання знань учнів було обрано новітню 100-бальну систему навчання, що останнім часом деякі школи почали впроваджувати в навчальний процес. Хоча таку систему і почали впроваджувати, проте наразі не було знайдено серед існуючих онлайн журналів такого, що б міг враховувати всі необхідні вимоги щодо оцінювання різних видів робіт. Тому було обрано створити більш модульну систему ведення журналу, яку кожна школа та кожен викладач міг-би адаптувати під себе.

З обов’язкових для всіх вимог фактично залишилася прив’язка журналу до навчального предмету, перелік яких надається та оновлюється за вимогами МОН. Це є необхідною мірою уніфікації для правильного функціонування системи профорієнтаційних рекомендацій.

На момент створення проекту повний список навчальних предметів згідно МОН складають такі предмети:

- Українська мова
- Українська література
- Зарубіжна література
- Іноземна мова
- Алгебра
- Геометрія
- Історія України

- Всесвітня історія
- Громадянська освіта
- Правознавство
- Економіка
- Біологія
- Географія
- Фізика
- Хімія
- Астрономія
- Мистецтво
- Трудове навчання. Технології
- Інформатика
- Основи здоров'я
- Фізична культура
- Захист України

Отож навчальний журнал було розділено на такі модулі:

- **Види роботи**: Тут вказуються види робіт, наперед визначена максимальна оцінка для певного виду роботи та їхня к-ть у навчальному році
- **Уроки**: Тут зберігаються відомості про урок як-то дата його проведення, тема заняття та домашнє завдання
- **Оцінки**: Тут зберігаються відомості про певну оцінку як-то її значення, урок у якому її виставили, тощо

Такий модульний розподіл складових журналу надає можливість в межах одного уроку оцінити учня за різними видами робіт, не обмежуючись кількістю клітинок, як це було у звичайних фізичних чи онлайн журналах.

Також варто завважити що для того, аби з кількох низьких оцінок не складалася краща оцінка, що могло б спантеличити при першому погляді користувача, у кожній клітинці навчальні успіхи учня відображаються, у форматі «к-ть набраних балів/максимальна к-ть балів» (наприклад 12/24) що однозначно вказує на те що це не висока оцінка за день, а на те, що тут є на що варто звернути увагу.

3.3.6 Створення початкових даних

На момент початку розробки в нас немає достовірних та повних даних для побудови на їх основі моделі штучного інтелекту, тому для першої ітерації більш-менш спроможної до передбачень моделі штучного інтелекту ми повинні створити базові синтетичні дані які надалі зможемо доповнювати вже реальними даними учнів та їх подальшим вибором спеціальності. Для побудови таких даних нам потрібно для кожної спеціальності розділити список всіх навчальних предметів на три пріоритети (високий, середній та низький) та обрати які показники тестів Айзенка та Голанда найбільше відповідають даній спеціальності.

Після численних перевірок та консультацій було створено наступний список спеціальностей і виявлені для них навчальні предмети та показники тестів відповідно до пріоритету:

Освіта/Педагогіка

Високий пріоритет: Українська мова, Історія України, Громадянська освіта, Іноземна мова, Українська література

Середній пріоритет: Всесвітня історія, Біологія, Географія, Алгебра, Геометрія

Низький пріоритет: Фізика, Хімія, Інформатика, Економіка, Астрономія, Мистецтво, Фізична культура, Трудове навчання. Технології, Основи здоров'я, Захист України, Правознавство

Пріоритетні показники тесту Айзенка: e, l

Пріоритетні показники тесту Голанда: s, c

Культура і мистецтво

Високий пріоритет: Мистецтво, Українська література, Зарубіжна література, Історія України, Всесвітня історія

Середній пріоритет: Іноземна мова, Громадянська освіта, Економіка, Географія, Українська мова

Низький пріоритет: Фізика, Хімія, Астрономія, Інформатика, Алгебра, Геометрія, Біологія, Фізична культура, Трудове навчання. Технології, Основи здоров'я, Захист України, Правознавство

Пріоритетні показники тесту Айзенка: l, e

Пріоритетні показники тесту Голанда: a, e

Гуманітарні науки

Високий пріоритет: Українська мова, Всесвітня історія, Іноземна мова, Громадянська освіта, Українська література

Середній пріоритет: Історія України, Економіка, Географія, Мистецтво, Зарубіжна література, Правознавство

Низький пріоритет: Фізика, Хімія, Астрономія, Інформатика, Алгебра, Геометрія, Біологія, Фізична культура, Трудове навчання. Технології, Основи здоров'я, Захист України

Пріоритетні показники тесту Айзенка: l, n

Пріоритетні показники тесту Голанда: i, c

Богослов'я

Високий пріоритет: Українська література, Історія України, Всесвітня історія, Мистецтво, Громадянська освіта

Середній пріоритет: Українська мова, Іноземна мова, Економіка, Основи здоров'я, Географія, Правознавство

Низький пріоритет: Трудове навчання. Технології, Фізика, Хімія, Біологія, Алгебра, Геометрія, Інформатика, Зарубіжна література, Фізична культура, Захист України, Астрономія

Пріоритетні показники тесту Айзенка: e, n

Пріоритетні показники тесту Голанда: s, a

Соціальні та поведінкові науки

Високий пріоритет: Громадянська освіта, Економіка, Історія України, Українська мова, Іноземна мова

Середній пріоритет: Всесвітня історія, Географія, Біологія, Українська література, Зарубіжна література, Правознавство

Низький пріоритет: Фізика, Хімія, Астрономія, Інформатика, Алгебра, Геометрія, Мистецтво, Фізична культура, Трудове навчання. Технології, Основи здоров'я, Захист України

Пріоритетні показники тесту Айзенка: e, l

Пріоритетні показники тесту Голанда: s, i

Журналістика

Високий пріоритет: Українська мова, Іноземна мова, Всесвітня історія, Українська література, Історія України

Середній пріоритет: Громадянська освіта, Мистецтво, Економіка, Зарубіжна література, Географія

Низький пріоритет: Фізика, Хімія, Астрономія, Інформатика, Алгебра, Геометрія, Біологія, Фізична культура, Трудове навчання. Технології, Основи здоров'я, Захист України, Правознавство

Пріоритетні показники тесту Айзенка: p, e

Пріоритетні показники тесту Голанда: s, c

Управління та адміністрування

Високий пріоритет: Економіка, Громадянська освіта, Інформатика, Українська мова, Трудове навчання. Технології, Правознавство

Середній пріоритет: Історія України, Всесвітня історія, Іноземна мова, Географія, Алгебра

Низький пріоритет: Геометрія, Фізика, Хімія, Біологія, Астрономія, Українська література, Зарубіжна література, Фізична культура, Основи здоров'я, Захист України, Мистецтво

Пріоритетні показники тесту Айзенка: e, c

Пріоритетні показники тесту Голанда: e, c

Право

Високий пріоритет: Громадянська освіта, Історія України, Українська мова, Всесвітня історія, Економіка, Правознавство

Середній пріоритет: Іноземна мова, Географія, Українська література, Зарубіжна література, Біологія

Низький пріоритет: Фізика, Хімія, Астрономія, Інформатика, Алгебра, Геометрія, Мистецтво, Фізична культура, Трудове навчання. Технології, Основи здоров'я, Захист України

Пріоритетні показники тесту Айзенка: e, c

Пріоритетні показники тесту Голанда: e, c

Біологія

Високий пріоритет: Біологія, Хімія, Географія, Фізика, Трудове навчання. Технології

Середній пріоритет: Економіка, Громадянська освіта, Астрономія, Алгебра, Українська мова

Низький пріоритет: Історія України, Всесвітня історія, Іноземна мова, Геометрія, Інформатика, Українська література, Зарубіжна література, Фізична культура, Основи здоров'я, Захист України, Мистецтво, Правознавство

Пріоритетні показники тесту Айзенка: n, i

Пріоритетні показники тесту Голанда: i, r

Природничі науки

Високий пріоритет: Біологія, Фізика, Хімія, Астрономія, Алгебра

Середній пріоритет: Геометрія, Географія, Інформатика, Українська мова, Іноземна мова

Низький пріоритет: Історія України, Всесвітня історія, Громадянська освіта, Українська література, Зарубіжна література, Економіка, Мистецтво, Фізична культура, Трудове навчання. Технології, Основи здоров'я, Захист України, Правознавство

Пріоритетні показники тесту Айзенка: n, i

Пріоритетні показники тесту Голанда: i, r

Математика та статистика

Високий пріоритет: Алгебра, Геометрія, Інформатика, Фізика, Економіка

Середній пріоритет: Астрономія, Іноземна мова, Українська мова, Біологія

Низький пріоритет: Історія України, Всесвітня історія, Громадянська освіта, Українська література, Зарубіжна література, Географія, Мистецтво, Фізична культура, Трудове навчання. Технології, Основи здоров'я, Захист України, Правознавство

Пріоритетні показники тесту Айзенка: i, n

Пріоритетні показники тесту Голанда: i, c

Інформаційні технології

Високий пріоритет: Інформатика, Алгебра, Фізика, Геометрія, Українська мова

Середній пріоритет: Економіка, Астрономія, Іноземна мова, Біологія

Низький пріоритет: Історія України, Всесвітня історія, Громадянська освіта, Українська література, Зарубіжна література, Географія, Мистецтво, Фізична культура, Трудове навчання. Технології, Основи здоров'я, Захист України, Правознавство

Пріоритетні показники тесту Айзенка: n, i

Пріоритетні показники тесту Голанда: r, i

Механічна інженерія

Високий пріоритет: Фізика, Алгебра, Геометрія, Трудове навчання. Технології, Інформатика

Середній пріоритет: Хімія, Астрономія, Економіка, Українська мова, Іноземна мова

Низький пріоритет: Історія України, Всесвітня історія, Громадянська освіта, Українська література, Зарубіжна література, Біологія, Географія, Мистецтво, Фізична культура, Основи здоров'я, Захист України, Правознавство

Пріоритетні показники тесту Айзенка: r, c

Пріоритетні показники тесту Голанда: r, i

Електрична інженерія

Високий пріоритет: Фізика, Алгебра, Геометрія, Інформатика, Трудове навчання. Технології

Середній пріоритет: Хімія, Астрономія, Економіка, Українська мова, Іноземна мова

Низький пріоритет: Історія України, Всесвітня історія, Громадянська освіта, Українська література, Зарубіжна література, Біологія, Географія, Мистецтво, Фізична культура, Основи здоров'я, Захист України, Правознавство

Пріоритетні показники тесту Айзенка: r, c

Пріоритетні показники тесту Голанда: r, i

Хімічна інженерія та біоінженерія

Високий пріоритет: Хімія, Фізика, Біологія, Алгебра, Геометрія

Середній пріоритет: Трудове навчання. Технології, Інформатика, Економіка, Громадянська освіта, Українська мова

Низький пріоритет: Історія України, Всесвітня історія, Іноземна мова, Астрономія, Географія, Українська література, Зарубіжна література, Фізична культура, Основи здоров'я, Захист України, Мистецтво, Правознавство

Пріоритетні показники тесту Айзенка: r, i

Пріоритетні показники тесту Голанда: i, r

Електроніка, автоматизація та електронні комунікації

Високий пріоритет: Фізика, Інформатика, Алгебра, Геометрія, Трудове навчання. Технології

Середній пріоритет: Астрономія, Економіка, Громадянська освіта, Хімія, Українська мова

Низький пріоритет: Історія України, Всесвітня історія, Іноземна мова, Біологія, Географія, Українська література, Зарубіжна література,

Фізична культура, Основи здоров'я, Захист України, Мистецтво, Правознавство

Пріоритетні показники тесту Айзенка: r, c

Пріоритетні показники тесту Голанда: r, i

Виробництво та технології

Високий пріоритет: Трудове навчання. Технології, Фізика, Алгебра, Геометрія, Інформатика

Середній пріоритет: Хімія, Економіка, Біологія, Громадянська освіта, Українська мова

Низький пріоритет: Історія України, Всесвітня історія, Іноземна мова, Астрономія, Географія, Українська література, Зарубіжна література, Фізична культура, Основи здоров'я, Захист України, Мистецтво, Правознавство

Пріоритетні показники тесту Айзенка: r, c

Пріоритетні показники тесту Голанда: r, i

Архітектура та будівництво

Високий пріоритет: Геометрія, Фізика, Алгебра, Трудове навчання. Технології, Мистецтво

Середній пріоритет: Економіка, Астрономія, Інформатика, Іноземна мова, Українська мова

Низький пріоритет: Історія України, Всесвітня історія, Громадянська освіта, Українська література, Зарубіжна література, Біологія, Географія, Фізична культура, Основи здоров'я, Захист України, Правознавство

Пріоритетні показники тесту Айзенка: a, p

Пріоритетні показники тесту Голанда: r, a

Аграрні науки та продовольство

Високий пріоритет: Біологія, Хімія, Географія, Трудове навчання. Технології, Українська мова

Середній пріоритет: Фізика, Економіка, Іноземна мова, Алгебра, Астрономія

Низький пріоритет: Історія України, Всесвітня історія, Громадянська освіта, Українська література, Зарубіжна література, Мистецтво, Інформатика, Геометрія, Фізична культура, Основи здоров'я, Захист України, Правознавство

Пріоритетні показники тесту Айзенка: a, r

Пріоритетні показники тесту Голанда: r, c

Ветеринарія

Високий пріоритет: Біологія, Хімія, Трудове навчання. Технології, Географія, Українська мова

Середній пріоритет: Фізика, Економіка, Іноземна мова, Астрономія, Алгебра

Низький пріоритет: Історія України, Всесвітня історія, Громадянська освіта, Українська література, Зарубіжна література, Мистецтво, Інформатика, Геометрія, Фізична культура, Основи здоров'я, Захист України, Правознавство

Пріоритетні показники тесту Айзенка: a, r

Пріоритетні показники тесту Голанда: r, c

Охорона здоров'я

Високий пріоритет: Біологія, Хімія, Фізика, Трудове навчання. Технології, Українська мова

Середній пріоритет: Економіка, Іноземна мова, Географія, Астрономія, Алгебра

Низький пріоритет: Історія України, Всесвітня історія, Громадянська освіта, Українська література, Зарубіжна література, Мистецтво, Інформатика, Геометрія, Фізична культура, Основи здоров'я, Захист України, Правознавство

Пріоритетні показники тесту Айзенка: a, r

Пріоритетні показники тесту Голанда: s, i

Соціальна робота

Високий пріоритет: Громадянська освіта, Українська мова, Історія України, Всесвітня історія, Іноземна мова

Середній пріоритет: Економіка, Мистецтво, Біологія, Географія, Українська література, Правознавство

Низький пріоритет: Фізика, Алгебра, Геометрія, Трудове навчання. Технології, Хімія, Інформатика, Зарубіжна література, Астрономія, Фізична культура, Основи здоров'я, Захист України

Пріоритетні показники тесту Айзенка: s, e

Пріоритетні показники тесту Голанда: s, c

Сфера обслуговування

Високий пріоритет: Економіка, Іноземна мова, Громадянська освіта, Українська мова, Трудове навчання. Технології

Середній пріоритет: Мистецтво, Біологія, Історія України, Всесвітня історія, Географія

Низький пріоритет: Фізика, Алгебра, Геометрія, Хімія, Українська література, Зарубіжна література, Астрономія, Інформатика, Фізична культура, Основи здоров'я, Захист України, Правознавство

Пріоритетні показники тесту Айзенка: e, p

Пріоритетні показники тесту Голанда: e, s

Воєнні науки, національна безпека, безпека державного кордону

Високий пріоритет: Захист України, Фізика, Громадянська освіта, Фізична культура, Географія

Середній пріоритет: Історія України, Всесвітня історія, Економіка, Українська мова, Трудове навчання. Технології, Правознавство

Низький пріоритет: Іноземна мова, Алгебра, Геометрія, Хімія, Біологія, Інформатика, Українська література, Зарубіжна література, Астрономія, Основи здоров'я, Мистецтво

Пріоритетні показники тесту Айзенка: r, p

Пріоритетні показники тесту Голанда: r, c

Цивільна безпека

Високий пріоритет: Громадянська освіта, Захист України, Фізика, Трудове навчання. Технології, Основи здоров'я, Правознавство

Середній пріоритет: Економіка, Історія України, Географія, Всесвітня історія, Фізична культура

Низький пріоритет: Алгебра, Геометрія, Хімія, Біологія, Іноземна мова, Інформатика, Українська мова, Зарубіжна література, Астрономія, Мистецтво, Українська література

Пріоритетні показники тесту Айзенка: r, s

Пріоритетні показники тесту Голанда: r, e

Транспорт

Високий пріоритет: Фізика, Трудове навчання. Технології, Алгебра, Геометрія, Інформатика

Середній пріоритет: Економіка, Астрономія, Іноземна мова, Українська мова, Хімія

Низький пріоритет: Історія України, Всесвітня історія, Громадянська освіта, Українська література, Зарубіжна література, Біологія, Географія, Мистецтво, Фізична культура, Основи здоров'я, Захист України, Правознавство

Пріоритетні показники тесту Айзенка: r, p

Пріоритетні показники тесту Голанда: r, i

Публічне управління та адміністрування

Високий пріоритет: Громадянська освіта, Економіка, Історія України, Українська мова, Іноземна мова, Правознавство

Середній пріоритет: Всесвітня історія, Алгебра, Геометрія, Мистецтво, Інформатика

Низький пріоритет: Фізика, Хімія, Біологія, Географія, Трудове навчання. Технології, Українська література, Зарубіжна література, Астрономія, Фізична культура, Основи здоров'я, Захист України

Пріоритетні показники тесту Айзенка: e, c

Пріоритетні показники тесту Голанда: e, c

Міжнародні відносини

Високий пріоритет: Іноземна мова, Громадянська освіта, Історія України, Всесвітня історія, Економіка, Правознавство

Середній пріоритет: Географія, Українська мова, Мистецтво, Українська література, Інформатика

Низький пріоритет: Фізика, Хімія, Біологія, Алгебра, Геометрія, Трудове навчання. Технології, Зарубіжна література, Астрономія, Фізична культура, Основи здоров'я, Захист України

Пріоритетні показники тесту Айзенка: e, n

Пріоритетні показники тесту Голанда: e, s

На основі цих списків було створено синтетичні дані за наступними правилами:

Примітка: Для можливості оцінювати успішність учня з певного предмету в будь-який момент часу – всі оцінки та максимально можливі оцінки по даному виду роботи просумовуються і приводяться до відсотку успішності учня з певного предмету.

Якщо у предмета високий пріоритет – відсоток успішності з нього генерується в межах від 90 до 100 відсотків.

Якщо у предмета середній пріоритет – в межах від 70 до 89 відсотків.

Якщо ж пріоритет низький – в межах від 50 до 69.

Для тесту Айзенка за його скороченою шкалою оцінювання максимальний бал обмежується 12 балами, тому розподіл був розроблений наступним чином:

Високий пріоритет: від 9 до 12 балів

Низький пріоритет: від 0 до 8 балів

У тесті Голанда максимально для певного критерію можна отримати до 40 балів, тому розподіл за пріоритетом для нього виглядає так:

Високий пріоритет: від 25 до 40 балів

Низький пріоритет: від 0 до 24 балів

В результаті ми отримуємо синтетичні дані для кожної спеціалізації які надалі можемо розширювати по мірі вступу учнів у університети.

3.3.7 Аналіз даних учнів

Як вже було вище описано, ми проводимо оцінку та передбачення спеціальності учня на основі 22-х предметів, 4 показників тесту Айзенка та 6 показників тесту Голанда.

По кожному предмету вираховується відсоткова успішність учня за певний період часу (для сумісності різних систем оцінювання)

Результати тестів Айзенка та Голанда аналізуються у їх відповідних межах:

Тест Айзенка: від 0 до 12 балів

Тест Голанда: від 0 до 40 балів

Якщо якихось даних не вистачає то вони не враховуються при передбаченні.

В результаті ми отримуємо передбачення на основі 32 показників. Система дає нам передбачення у вигляді списку з п’яти спеціальностей, що можуть зацікавити учня, базуючись на його успішності в навчанні та результатах психологічних тестувань.

3.4 Перелік бібліотек використаних під час розробки додатку

Під час реалізації бекенд частини були використані такі бібліотеки:

Spring Boot пропонує спеціальні залежності, відомі як стартери (starters), які забезпечують швидку інтеграцію необхідних функцій і автоматичну конфігурацію. Стартери значно спрощують розробку, зменшуючи кількість ручних налаштувань і дозволяючи зосередитися на бізнес-логіці. До таких бібліотек відносяться:

- **spring-boot-starter-web**:
  Ця бібліотека забезпечує основу для розробки веб-додатків у середовищі Spring Boot, включаючи підтримку архітектури RESTful та інтеграцію з Spring MVC. Starter включає залежності для роботи з HTTP-протоколом, маршрутизації запитів, серіалізації об'єктів і вбудованих серверів, таких як Tomcat або Jetty.

- **spring-boot-starter-data-jpa**:
  Ця бібліотека забезпечує інтеграцію з реляційними базами даних через стандарт JPA (Java Persistence API), автоматизуючи процес ORM. Starter використовує Hibernate як реалізацію JPA, включає засоби для створення репозиторіїв і виконання запитів. Дозволяє зменшити обсяг конфігурацій завдяки автоматичному налаштуванню.

- **spring-boot-starter-data-redis**:
  Бібліотека забезпечує інтеграцію з Redis, використовуючи Spring Data Redis. Вона підтримує роботу з ключ-значення сховищем, автоматичну конфігурацію з'єднання, управління кешем, публікацію-підписку

(Pub/Sub) та роботу з транзакціями. Starter значно спрощує інтеграцію Redis у проєкти Spring Boot.

- **spring-boot-starter-security**:
  Ця бібліотека надає автоматичну конфігурацію для інтеграції Spring Security у Spring Boot-додатки. Вона забезпечує механізми автентифікації, авторизації, захисту від атак CSRF та налаштування безпечного доступу до REST API. Starter дозволяє швидко застосовувати сучасні стандарти безпеки.

Окрім стартерів, для вирішення специфічних задач були використані наступні бібліотеки:

- **flyway-core**:
  Flyway Core — інструмент для управління міграціями баз даних. Він дозволяє версіонувати структуру БД за допомогою SQL або Java-скриптів, забезпечуючи автоматичне застосування змін при оновленні програми. Бібліотека інтегрується з більшістю реляційних баз даних і легко адаптується до середовищ CI/CD.

- **spring-boot-devtools**:
  Ця бібліотека надає інструменти для покращення процесу розробки, включаючи автоматичне перезавантаження додатка при зміні коду, спрощене підключення до віддалених серверів і вимкнення кешу у режимі розробки. DevTools підвищує продуктивність розробника, забезпечуючи швидкий зворотний зв'язок під час тестування.

- **postgresql**:
  Ця бібліотека є JDBC-драйвером для роботи з PostgreSQL — потужною реляційною базою даних. Вона дозволяє виконувати SQL-запити, інтегрується з інструментами ORM, такими як Hibernate, і забезпечує підтримку специфічних можливостей PostgreSQL, таких як JSON, масиви, CTE-запити та розширення.

- **jjwt-impl**:

  Бібліотека надає основні компоненти для створення, підпису, валідації та аналізу JSON Web Tokens (JWT). Вона підтримує симетричні та асиметричні алгоритми підпису, забезпечуючи стандартизовану автентифікацію та передачу даних між сторонами з підтвердженням їх цілісності.

- **jjwt-api**:

  Ця бібліотека надає API для роботи з JSON Web Tokens (JWT), визначаючи інтерфейси і абстракції для створення, підпису, валідації та декодування токенів. Вона забезпечує зручний і стандартизований підхід для автентифікації, передачі даних та забезпечення їх цілісності в розподілених системах.

- **jjwt-jackson**:

  Ця бібліотека є розширенням для jjwt-api, що використовує Jackson для серіалізації та десеріалізації JSON. Вона інтегрується з екосистемою Jackson для роботи з кастомними моделями даних, забезпечуючи гнучке та ефективне управління JWT-змістом у JSON-форматі.

- **kotlinx-coroutines-core**:

  Основна бібліотека для роботи з корутинами в Kotlin. Вона забезпечує підтримку асинхронного програмування через легкі потоки, дозволяючи працювати з конкурентністю без блокування. Бібліотека включає механізми для запуску, скасування, синхронізації завдань та управління контекстами корутин.

- **kotlin-reflect**:

  Бібліотека Kotlin Reflection надає інструменти для роботи з метаданими класів, методів, властивостей і анотацій під час виконання програми. Вона дозволяє виконувати динамічний аналіз типів, викликати методи

або отримувати значення властивостей без жорсткої прив’язки до конкретних класів.

- **icu4j**:
  Модуль ICU4J (International Components for Unicode) надає широкий набір інструментів для роботи з локалізацією, обробкою тексту, форматуванням чисел, дат, валют та підтримкою Unicode. Бібліотека розширює стандартні можливості Java для міжнародної підтримки додатків, орієнтованих на різноманітні культури й мови.

- **h2o-core**:
  Ця бібліотека є ядром платформи H2O.ai і забезпечує базову функціональність для роботи з моделями машинного навчання. Вона включає інструменти для попередньої обробки даних, виконання розподілених обчислень і інтеграції з іншими компонентами H2O.

- **h2o-genmodel**:
  Ця бібліотека спеціально розроблена для інтеграції моделей H2O, збережених у форматі MOJO, у сторонні програми. Вона надає інструменти для завантаження, виконання та інференсу таких моделей у продакшн-середовищі. Бібліотека дозволяє використовувати MOJO-моделі без необхідності запуску сервера H2O, що забезпечує високу продуктивність і простоту інтеграції в додатках на JVM.

Під час реалізації фронтенд-частини були використані наступні бібліотеки:

Фреймворк **React** є основою для побудови інтерфейсу користувача, забезпечуючи компонентно-орієнтовану архітектуру, яка спрощує розробку динамічних та інтерактивних веб-додатків. У поєднанні з React використовувалися додаткові бібліотеки для розширення функціоналу:

- **react-dom**: бібліотека для рендерингу React-компонентів у DOM, яка забезпечує інтеграцію між компонентами React та браузером.

- **react-router-dom**: забезпечує маршрутизацію у веб-додатку, дозволяючи створювати багатосторінкову навігацію без перезавантаження сторінки.

Для створення сучасного й стильного інтерфейсу використовувався фреймворк **Material-UI (MUI)**, що включає наступні компоненти:

- **@mui/material**: основна бібліотека компонентів, що забезпечує готові рішення для побудови зручного інтерфейсу користувача.
- **@mui/icons-material**: набір іконок для використання в компонентах MUI, який відповідає сучасним стандартам дизайну.
- **@emotion/styled**: бібліотека для стилізації компонентів, що інтегрується з MUI, забезпечуючи простоту написання CSS-стилів.

Для роботи з cookies та збереження сесійної інформації застосовувалась:

- **js-cookie**: бібліотека для зручної роботи з cookies у браузері, що дозволяє ефективно зберігати та отримувати дані сесій.

Для валідації типів пропсів у компонентах React використовувалася:

- **prop-types**: бібліотека для перевірки типів переданих в компонент властивостей, що забезпечує додатковий рівень надійності та зручності в розробці.

Додатково використовувався **Bootstrap**, популярний CSS-фреймворк, який забезпечує набір готових стилів і компонентів для створення адаптивного, естетичного дизайну. Його використання дозволило швидко реалізувати гнучкий інтерфейс із дотриманням принципів сучасного веб-дизайну.

Ця комбінація інструментів дозволила створити функціональний, гнучкий і стильний інтерфейс користувача з мінімальними зусиллями.

Висновок до розділу 3

У ході виконання 3 розділу було ретельно розглянуто та вирішено багато ключових завдань, спрямованих на розробку фінального продукту.

Велика увага була приділена структурним особливостям реалізації деяких елементів системи, які вимагали структурних змін внаслідок адаптації до 200-бальної системи оцінювання та інтеграції у систему для узагальнення результатів навчання функції профорієнтації за допомогою штучного інтелекту.

ВИСНОВКИ

У рамках даної роботи було розглянуто актуальність інформаційної системи для узагальнення результатів навчання з функцією профорієнтації в умовах сьогодення. Також було розроблено кросплатформений вебдодаток, що реалізовує функціонал по збереженню результатів навчання учнів, реалізовує функціонал по тестуванню учнів за тестами Айзенка та Голанда, вирішує проблему відображення кількох отриманих оцінок в межах одного уроку та має можливість проаналізувати результати успішності учня та його психологічних тестів та надати рекомендації по майбутній профорієнтації при вступі на основі цих даних.

Розробка та впровадження інформаційної системи є кроком у напрямку забезпечення індивідуального підходу до кожного учня, надання об'єктивних рекомендацій щодо найбільш відповідних напрямків навчання та професій, а також підтримки їхнього самовизначення та розвитку.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Шкіль Л. Тільки кожен шостий студент закінчує IT-курси до кінця – дослідження. *AIN.UA*. URL: https://ain.ua/2023/01/05/tilky-kozhen-shostyj-student-zakinchuye-it-kursy-do-kinczya-doslidzhennya/ (дата звернення: 13.09.2024).
2. Wilton N. Introduction to human resource management. 5-те вид. New York : SAGE Publications, Limited, 2022. 552 с.
3. Gupta A. D. Strategic human resource management. New York, NY : Routledge, 2020. : Productivity Press, 2020. URL: https://doi.org/10.4324/9780429327728 (дата звернення: 27.09.2024).
4. Юрченко О. Електронний журнал у школі: як перейти і з чого розпочати?. *Освіторія Медіа*. URL: https://osvitoria.media/experience/elektronnyj-zhurnal-u-shkoli-yak-perejty-i-z-chogo-rozpochaty/ (дата звернення: 16.09.2024).
5. Методичні рекомендації щодо запровадження в громаді цифрового інструменту “електронні щоденники”. *hromada.gov.ua*. URL: https://backend.hromada.gov.ua/storage/uploads/images/project/elektronni-shhodenniki/Методичні%20рекомендації%20е-щоденники.pdf (дата звернення: 17.09.2024).
6. Перепелиця А. С., Бабаков Р. М. Інформаційна система для узагальнення результатів навчання з функцією профорієнтації. *Прикладні аспекти сучасних міждисциплінарних досліджень* : Всеукр. науково-практ. конф., м. Вінниця, 1 листоп. 2024 р. Вінниця, 2024. URL: https://jpasmd.donnu.edu.ua (дата звернення: 20.09.2024).
7. Електронні класні журнали та щоденники з можливостями дистанційного навчання. *Нові знання*. URL: https://nz.ua (дата звернення: 16.09.2024).
8. Державні безкоштовні електронні щоденники та журнали для закладів загальної середньої освіти. *E-Journal*. URL: https://e-journal.iea.gov.ua (дата звернення: 16.09.2024).
9. Розділ "Електронні журнали". *Всеосвіта*. URL: https://vseosvita.ua/journal/2023 (дата звернення: 18.09.2024).

10. Електронний журнал та щоденник. *Atoms.ua*. URL: https://atoms.ua (дата звернення: 18.09.2024).
11. Kotlin docs. *Kotlin Documentation*. URL: https://kotlinlang.org/docs/home.html (дата звернення: 18.09.2024).
12. Spring Boot. *Spring | Home*. URL: https://docs.spring.io/spring-boot/index.html (дата звернення: 20.09.2024).
13. Миколайчук В. Kotlin для Java-розробника: варто чи ні?. *Dou.ua*. URL: https://dou.ua/forums/topic/34087/ (дата звернення: 20.09.2024).
14. Silberschatz A., Korth H. F., Sudarshan S. Database system concepts : навч. посіб. 7-ме вид. McGraw-Hill Education, 2019. 1376 с. URL: https://db-book.com (дата звернення: 20.09.2024).
15. Momjian B. PostgreSQL: introduction and concepts. Pearson Education, 2000. 462 с.
16. PostgreSQL 17.2 documentation. *PostgreSQL Documentation*. URL: https://www.postgresql.org/docs/current/index.html (дата звернення: 20.09.2024).
17. Порівняння СУБД My SQL, PostgreSQL та MS SQL Server. *Курси бізнес аналітики | Україна | DataBI*. URL: https://data-b-i.com/uk/article/porivnyannya-subd-mysql-postgresql-mssqlserver.html (дата звернення: 20.09.2024).
18. Денисенко А. Битва титанів: що краще – PostgreSQL чи MySQL?. *Highload.today - медіа для розробників*. URL: https://highload.today/uk/postgresql-vs-mysql/ (дата звернення: 20.09.2024).
19. Рудрамані П. Різниця між SQLite та MySQL. *gocoding.org*. URL: https://gocoding.org/uk/різниця-між-sqlite-i-mysql/ (дата звернення: 19.09.2024).
20. SQLite advantages and disadvantages - javatpoint. *www.javatpoint.com*. URL: https://www.javatpoint.com/sqlite-advantages-and-disadvantages (дата звернення: 19.09.2024).
21. Hitesh J. MariaDB vs MYSQL - what's the difference? (pros and cons). *Cloud Infrastructure Services*. URL: https://cloudinfrastructureservices.co.uk/mariadb-vs-mysql/ (дата звернення: 18.11.2024).

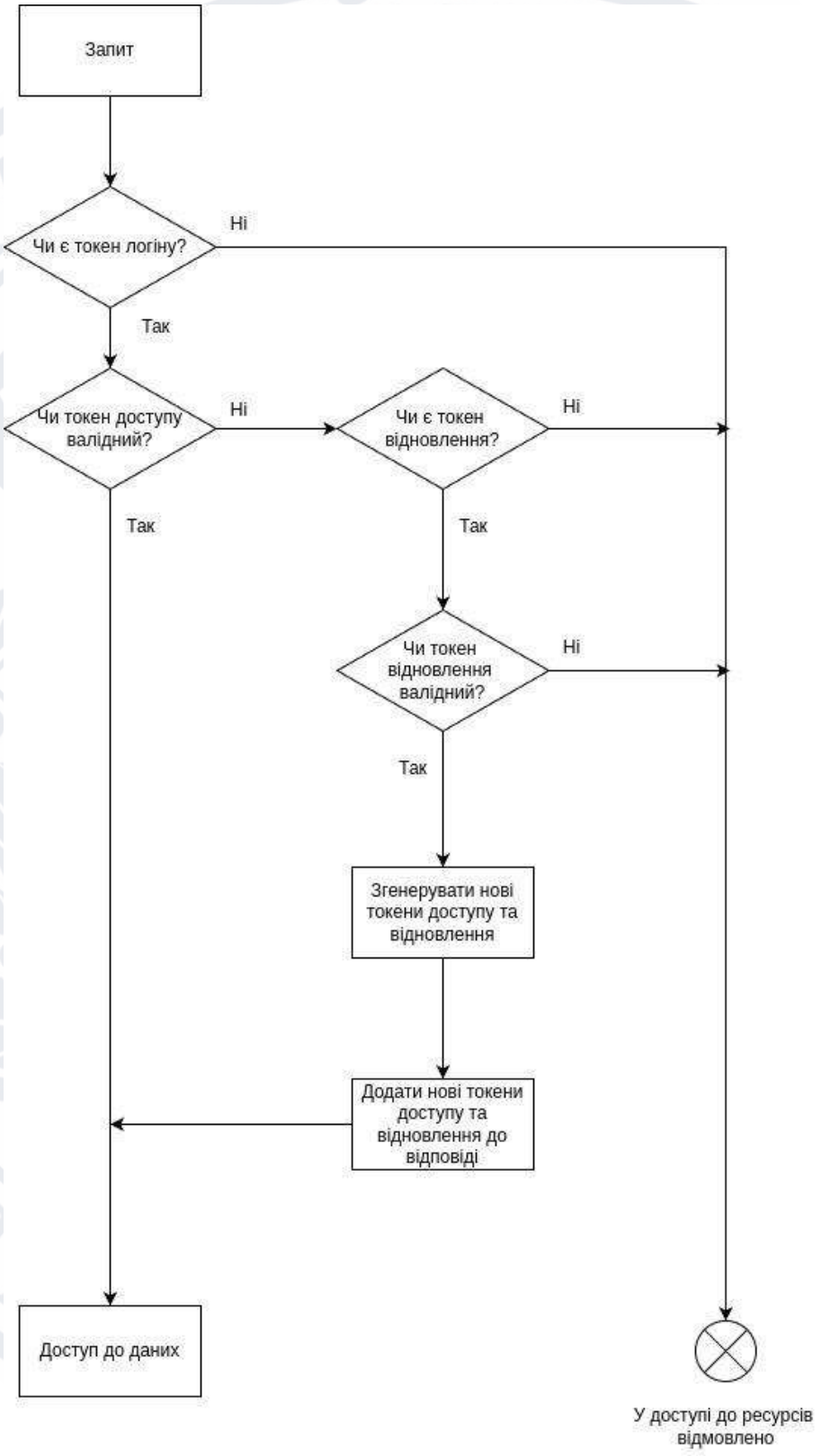
22. MariaDB vs MySQL - A detailed comparison in 2024. *RunCloud Blog*. URL: https://runcloud.io/blog/mariadb-vs-mysql (дата звернення: 18.09.2024).
23. Kaczorowski C. A comparative analysis of contemporary integrated java environments. *Journal of computer sciences institute*. 2023. Т. 26. С. 42–47. URL: https://doi.org/10.35784/jcsi.3077 (дата звернення: 29.11.2024).
24. VS Code vs. WebStorm – a detailed comparison. *Swimm: AI Code Documentation And Knowledge Sharing*. URL: https://swimm.io/blog/vscode-vs-webstorm-a-detailed-comparison (дата звернення: 23.09.2024).
25. Ibrahim M. H., Sayagh M., Hassan A. E. A study of how Docker Compose is used to compose multi-component systems. *Empirical software engineering*. 2021. Т. 26, № 6. URL: https://doi.org/10.1007/s10664-021-10025-1 (дата звернення: 29.11.2024).
26. Gradle vs Maven comparison. *Gradle*. URL: https://gradle.org/maven-vs-gradle/ (дата звернення: 20.09.2024).
27. Bhabishya, Gurung. Theseus: A comparative analysis of create-react-app (CRA) and Vite for React.js projects. *Ammattikorkeakoulut - Theseus*. URL: https://www.theseus.fi/handle/10024/860241 (дата звернення: 24.09.2024).
28. React - JavaScript-бібліотека для створення користувацьких інтерфейсів. *React*. URL: https://react.dev (дата звернення: 19.09.2024).
29. Сучасний підручник з JavaScript. *Сучасний підручник з JavaScript*. URL: https://uk.javascript.info (дата звернення: 25.09.2024).
30. Bootstrap · The most popular HTML, CSS, and JS library in the world. *Bootstrap*. URL: https://getbootstrap.com (дата звернення: 23.09.2024).
31. MUI: The React component library you always wanted. *MUI.com*. URL: https://mui.com (дата звернення: 23.09.2024).
32. Apache Mahout. *Apache Mahout*. URL: https://mahout.apache.org (дата звернення: 01.10.2024).
33. MLlib | Apache Spark. *Apache Spark™ - Unified Engine for large-scale data analytics*. URL: https://spark.apache.org/mllib/ (дата звернення: 01.10.2024).

34. Convergence of the world's best predictive & generative AI. *H2O.ai*. URL: https://h2o.ai (дата звернення: 01.10.2024).
35. Using Flow - H2O's Web UI – H2O 3.46.0.6 documentation. *H2O.ai Documentation*. URL: https://docs.h2o.ai/h2o/latest-stable/h2o-docs/flow.html (дата звернення: 01.10.2024).
36. Опитувальник професійної спрямованості (ОПС) Д. голанда. *Тернопільський обласний центр зайнятості*. URL: https://ter.dcz.gov.ua/publikaciya/opytuvalnyk-profesiynoyi-spryamovanosti-ops-d-golanda (дата звернення: 02.10.2024).
37. Ларін Д. І. Тести професійної орієнтації учнів старших класів та професійний відбір: методики, опитувальники. *Всеосвіта*. URL: https://vseosvita.ua/library/testi-profesijnoi-orientacii-ucniv-starsih-klasiv-ta-profesijnij-vidbir-metodiki-opituvalniki-23396.html (дата звернення: 20.09.2024).
38. Особистісний опитувальник Айзенка, EPQ/PEN. *Testoteka.info*. URL: https://testoteka.info/uk/test/typo/eysenck-epq (дата звернення: 01.10.2024).
39. Тест Амтхауера. *The Amthauer test*. URL: https://amthauer-ist.com/ua/ (дата звернення: 24.09.2024).
40. Тест структури інтелекту Р. Амтхауера. *Stud.com.ua*. URL: https://stud.com.ua/62246/psihologiya/test_strukturi_intelektu_amthauera (дата звернення: 23.09.2024).
41. Тест Кеттелла. *ПСИХОЛОГІС - енциклопедія практичної психології*. URL: http://psychologis.com.ua/test_kettella.htm (дата звернення: 23.09.2024).
42. Барієва А. О. Тест Кеттелла (16 RF-опитувальник). *Всеосвіта*. URL: https://vseosvita.ua/test/test-kettella-16-rf-opytuvalnyk-1691947.html (дата звернення: 25.09.2024).
43. Тест Люшера. *Дитячий психолог*. URL: https://dytpsyholog.com/2015/04/04/тест-люшера/ (дата звернення: 23.09.2024).
44. Тест Люшера. Інтерпретація методики. *Всеосвіта*. URL: https://vseosvita.ua/c/news/post/72445 (дата звернення: 23.09.2024).

45. Методика діагностики особистості на мотивацію до успіху Т. Елерса. *Stud.com.ua*. URL: https://stud.com.ua/17007/menedzhment/metodika_diagnostiki_osobistosti_motivatsiyu_uspihu_elersa (дата звернення: 23.09.2024).
46. Методика "Карта інтересів". *Proforientator.info*. URL: https://proforientator.info/?page_id=6006 (дата звернення: 07.10.2024).
47. Карта інтересів (модифікована методика А.Е. Голомштока). *Stud.com.ua*. URL: https://studfile.net/preview/5393985/page:6/ (дата звернення: 07.10.2024).
48. Перепелиця А. С., Богач І. В. Переваги використання фреймворку Spring при розробці веб-додатків. *Прикладні аспекти сучасних міждисциплінарних досліджень* : Матеріали Міжн. наук.-практ. конф, м. Вінниця, 15 груд. 2022 р. Вінниця, 2022. С. 163–165.
49. Figma: the collaborative interface design tool. *Figma.com*. URL: https://www.figma.com (дата звернення: 23.09.2024).
50. Редакція. Покроковий посібник по роботі в Figma. Урок зі створення мобільного додатка. *UXPUB Українська дизайн-спільнота*. URL: https://ux.pub/editorial/pokrokovie-kierivnitstvo-po-roboti-v-figma-urok-zi-stvoriennia-mobilnogho-dodatka-53a9 (дата звернення: 30.09.2024).
51. Як використовувати JSON Web Tokens (JWT) для автентифікації. *DevZone*. URL: https://devzone.org.ua/post/iak-vykorystovuvaty-json-web-tokens-jwt-dlia-avtentyfikatsiyi (дата звернення: 07.10.2024).
52. Яке призначення атрибута "httpOnly" у файлах cookie HTTP?. *Європейська інформаційна технологія сертифікаційна академія*. URL: https://uk.eitca.org/cybersecurity/eitc-is-wapt-web-applications-penetration-testing/web-attacks-practice/http-attributes-cookie-stealing/examination-review-http-attributes-cookie-stealing/what-is-the-purpose-of-the-httponly-attribute-in-http-cookies/ (дата звернення: 17.10.2024).

ДОДАТОК А

Блок-схема системи перевірки доступу користувача до ресурсів



ДОДАТОК Б

--- Вміст файлу: ./djournal/JournalApplication.java ---

```
package com.djournal;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class JournalApplication {

    public static void main(String[] args) {
        SpringApplication.run(JournalApplication.class, args);
    }

}
```

--- Вміст файлу: ./djournal/configuration/BeanConfiguration.kt ---

```
package com.djournal.configuration

import hex.genmodel.MojoModel
import hex.genmodel.easy.EasyPredictModelWrapper
import io.lettuce.core.RedisClient
import io.lettuce.core.RedisURI
import lombok.RequiredArgsConstructor
import org.springframework.boot.autoconfigure.data.redis.RedisProperties
import org.springframework.context.annotation.Bean
import org.springframework.context.annotation.Configuration
import
org.springframework.security.config.annotation.method.configuration.EnableMeth
odSecurity
import org.springframework.web.servlet.config.annotation.CorsRegistry
import
org.springframework.web.servlet.config.annotation.WebMvcConfigurer

@Configuration
@RequiredArgsConstructor
@EnableMethodSecurity
class BeanConfiguration(
```

```
    private val redisProperties: RedisProperties
) {


    @Bean
    fun redisClient(): RedisClient {
        return RedisClient.create(RedisURI.create(redisProperties.host,
redisProperties.port))
    }

    @Bean
    fun corsConfigurer(): WebMvcConfigurer {
        return object : WebMvcConfigurer {
            override fun addCorsMappings(registry: CorsRegistry) {
                registry.addMapping("/**")
            }
        }
    }

    @Bean
    fun predictModelWrapper() : EasyPredictModelWrapper {
        return
EasyPredictModelWrapper(MojoModel.load("./predict_model.mojo"))
    }
}

--- Вміст файлу: ./djournal/controller/AdminController.java ---
package com.djournal.controller;

import org.springframework.web.bind.annotation.RestController;

@RestController
public class AdminController {
}

--- Вміст файлу: ./djournal/controller/AuthenticationController.java ---
package com.djournal.controller;
```

```
import com.djournal.service.authentication.UserAuthenticationService;
import lombok.RequiredArgsConstructor;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
@RequestMapping("/auth")
@RequiredArgsConstructor
public class AuthenticationController {

    private final UserAuthenticationService service;


//    @PostMapping("/login")
//    public ResponseEntity<TokensDto> login(@RequestBody
LoginPasswordDto loginPasswordDto)
//    {
//        return
ResponseEntity.ok().body(service.authenticateByLogin(loginPasswordDto));
//    }
}

--- Вміст файлу: ./djournal/controller/ClassesController.kt ---
package com.djournal.controller

import com.djournal.domain.Classes
import com.djournal.service.ClassesService
import org.springframework.dao.DataIntegrityViolationException
import org.springframework.http.HttpStatus
import org.springframework.http.ResponseEntity
import org.springframework.web.bind.annotation.ExceptionHandler
import org.springframework.web.bind.annotation.GetMapping
import org.springframework.web.bind.annotation.PostMapping
import org.springframework.web.bind.annotation.RequestBody
import org.springframework.web.bind.annotation.RequestMapping
import org.springframework.web.bind.annotation.RestController

@RestController
```

```
@RequestMapping("/classes")
class ClassesController(
    private val classesService: ClassesService
) {
    @GetMapping
    fun getClasses(): ResponseEntity<List<Classes>> {
        return ResponseEntity.ok(classesService.getClasses())
    }

    @PostMapping
    fun saveClassData(@RequestBody classData: Classes):
ResponseEntity<Unit> {
        classesService.saveClass(classData)
        return ResponseEntity.ok().build()
    }

    @ExceptionHandler(DataIntegrityViolationException::class)
    fun classAlreadyExistExceptionHandler(e:
DataIntegrityViolationException) : ResponseEntity<String>
    {
        return ResponseEntity.status(HttpStatus.BAD_REQUEST).body("Клас
з таким ім\'ям вже існує")
    }
}

--- Вміст файлу: ./djournal/controller/JournalController.kt ---
package com.djournal.controller

import com.djournal.domain.Lesson
import com.djournal.domain.dto.JournalDTO
import com.djournal.domain.dto.LessonMarksDto
import com.djournal.service.JournalService
import com.djournal.service.LessonService
import com.djournal.service.MarkService
import lombok.RequiredArgsConstructor
import org.springframework.http.ResponseEntity
import org.springframework.web.bind.annotation.*
import java.security.Principal
```

```
import kotlin.jvm.optionals.getOrNull

@RestController
@RequiredArgsConstructor
@RequestMapping("/journal")
class JournalController(
    private val service: JournalService,
    private val lessonService: LessonService,
    private val markService: MarkService,
) {

    @GetMapping("/all/teacher")
    fun getJournalsByTeacher(userId: Principal):
ResponseEntity<List<JournalDTO>> {
        return
ResponseEntity.ok(service.findAllByTeacherId(userId.name!!.toInt()).map {
it.toDTO() })
    }
    @GetMapping("/all/student")
    fun getJournalsByStudent(userId: Principal):
ResponseEntity<List<JournalDTO>> {
        return
ResponseEntity.ok(service.findAllByStudentsId(userId.name!!.toInt()).map {
it.toDTO() })
    }

    @GetMapping("/{id}")
    fun getJournalById(
        @PathVariable id: Int,
    ): ResponseEntity<JournalDTO> {
        return
ResponseEntity.ok(service.findOneById(id).getOrNull()?.toDTO())
    }

    @GetMapping("/{id}/lessons")
    fun getLessonsByJournalId(@PathVariable id: Int):
ResponseEntity<List<Lesson>> {
        val journal = service.findOneById(id).getOrNull()
```

```
            return ResponseEntity.ok(journal?.lessons)
        }

        @GetMapping("/{id}/lessons/{month}")
        fun getLessonsByJournalIdFilterByMonth(@PathVariable id: Int,
@PathVariable month: Int): ResponseEntity<List<Lesson>> {
            val journal = service.findOneById(id).getOrNull()
            return ResponseEntity.ok(journal?.lessons?.filter { lesson ->
lesson.date.monthValue == month })
        }

        @PostMapping(value = ["/mark"])
        fun saveLessonMarks(@RequestBody lessonMarks: LessonMarksDto):
ResponseEntity<Lesson?> {
            markService.saveMarksByLesson(lessonMarks)
            return ResponseEntity.ok(lessonService.findById(lessonMarks.lesson))
        }
    }

    --- Вміст файлу: ./djournal/controller/LessonController.kt ---
    package com.djournal.controller

    import com.djournal.domain.Lesson
    import com.djournal.domain.Mark
    import com.djournal.domain.dto.LessonEditDTO
    import com.djournal.service.LessonService
    import org.springframework.http.ResponseEntity
    import org.springframework.web.bind.annotation.*

    @RestController
    @RequestMapping("/lesson")
    class LessonController(
        private val lessonService: LessonService,
    ) {
        @GetMapping("/{lessonId}/marks")
        fun getMarksByJournalId(@PathVariable lessonId: Int):
ResponseEntity<List<Mark>?> {
            val lesson: Lesson? = lessonService.findById(lessonId)
```

```
            if (lesson != null) {
                val marks = lesson.marks
                return ResponseEntity.ok(marks)
            } else {
                return ResponseEntity.badRequest().build()
            }
        }

        @PostMapping("/save")
        fun saveLesson(@RequestBody lessonDTO: LessonEditDTO):
ResponseEntity<Lesson> {
            println(lessonDTO)
            val lesson: Lesson
            if (lessonDTO.id != null && lessonDTO.id != 0) {
                val oldLesson = lessonService.findById(lessonDTO.id) ?: return
ResponseEntity.badRequest().build()
                lesson = oldLesson.copy(
                    date = lessonDTO.date,
                    journal = lessonDTO.journal,
                    lessonNumber = lessonDTO.lessonNumber,
                    theme = lessonDTO.theme,
                    homeWork = lessonDTO.homeWork
                )
            } else {
                lesson = Lesson(
                    date = lessonDTO.date,
                    journal = lessonDTO.journal,
                    lessonNumber = lessonDTO.lessonNumber,
                    theme = lessonDTO.theme,
                    homeWork = lessonDTO.homeWork
                )
            }

            return ResponseEntity.ok(lessonService.save(lesson))
        }

        @PostMapping("/delete/{id}")
        fun deleteLesson(@PathVariable id: Int): ResponseEntity<Any> {
```

```
        lessonService.deleteById(id)
        return ResponseEntity.ok().build()
    }

}
--- Вміст файлу: ./djournal/controller/MarkController.kt ---
package com.djournal.controller

import com.djournal.domain.Lesson
import com.djournal.domain.dto.LessonMarksDto
import com.djournal.service.LessonService
import com.djournal.service.MarkService
import org.springframework.http.ResponseEntity
import org.springframework.web.bind.annotation.PostMapping
import org.springframework.web.bind.annotation.RequestBody
import org.springframework.web.bind.annotation.RequestMapping
import org.springframework.web.bind.annotation.RestController

@RestController
@RequestMapping("/mark")
class MarkController(
    private val markService: MarkService,
    private val lessonService: LessonService
) {
    @PostMapping("/save-all")
    fun saveLessonMarks(@RequestBody lessonMarks: LessonMarksDto):
ResponseEntity<Lesson?>? {
        markService.saveMarksByLesson(lessonMarks)
        return ResponseEntity.ok(lessonService.findById(lessonMarks.lesson))
    }
}
--- Вміст файлу: ./djournal/controller/OwnerController.java ---
package com.djournal.controller;

import org.springframework.web.bind.annotation.RestController;

@RestController
public class OwnerController {
```

```
}

--- Вміст файлу: ./djournal/controller/PredictionController.kt ---
package com.djournal.controller

import com.djournal.domain.dto.TestCompletionDTO
import com.djournal.service.PredictionService
import org.springframework.http.ResponseEntity
import org.springframework.web.bind.annotation.GetMapping
import org.springframework.web.bind.annotation.RequestMapping
import org.springframework.web.bind.annotation.RestController
import java.security.Principal

@RestController
@RequestMapping("/prediction")
class PredictionController(
    var predictionService: PredictionService,
) {
    @GetMapping
    fun predict(userId: Principal) : ResponseEntity<List<String>>
    {
        val prediction = predictionService.predictJobs(userId.name.toInt())
        return ResponseEntity.ok(prediction)
    }
}
--- Вміст файлу: ./djournal/controller/RestLoginController.kt ---
package com.djournal.controller

import com.djournal.domain.LoginState
import com.djournal.domain.User
import com.djournal.domain.dto.LoginDTO
import com.djournal.service.LoginService
import com.djournal.util.HTTPUtil
import jakarta.servlet.http.HttpServletResponse
import org.springframework.http.HttpStatus
import org.springframework.http.ResponseEntity
import org.springframework.web.bind.annotation.*
```

```
@RestController
@RequestMapping("/login")
class RestLoginController(
    val loginService: LoginService
) {
    @PostMapping
    fun login(
        @RequestBody loginDTO: LoginDTO,
        response: HttpServletResponse
    ): ResponseEntity<Any> {
        val loginStatus =
loginService.loginByUsernameAndPassword(response, loginDTO.username,
loginDTO.password)
        if (loginStatus == LoginState.NOT_AUTHENTICATED) {
            return
ResponseEntity.status(loginStatus.getResponseStatus()).body("Username or
password is incorrect")
        }
        return ResponseEntity.status(loginStatus.getResponseStatus()).build()
    }

    @GetMapping("/ping")
    fun ping(): ResponseEntity<Any> {
        return ResponseEntity.status(HttpStatus.OK).build()
    }

    @PostMapping("/logout")
    fun logout(response: HttpServletResponse): ResponseEntity<Any> {
        response.reset()
        response.addCookie(HTTPUtil.createPublicCookie("Authenticated",
"False"))
        response.addCookie(HTTPUtil.suspendCookie("token"))
        response.addCookie(HTTPUtil.suspendCookie("refresh-token"))
        response.addCookie(HTTPUtil.suspendCookie("role", false))
        return ResponseEntity.ok().build()
    }
}
```

--- Вміст файлу: ./djournal/controller/StudentController.kt ---

```
package com.djournal.controller

import com.djournal.domain.Lesson
import com.djournal.service.JournalService
import org.springframework.http.ResponseEntity
import org.springframework.security.access.prepost.PreAuthorize
import org.springframework.security.core.Authentication
import org.springframework.web.bind.annotation.GetMapping
import org.springframework.web.bind.annotation.PathVariable
import org.springframework.web.bind.annotation.RequestMapping
import org.springframework.web.bind.annotation.RestController
import kotlin.jvm.optionals.getOrNull

@RestController
@RequestMapping("student")
class StudentController(
    private val journalService: JournalService,
) {
//    @PreAuthorize("hasAuthority('CLIENT')")
    @GetMapping("/{id}/lessons/{month}")
    fun getLessonsByJournalIdFilterByMonthAndByUser(
        @PathVariable id: Int,
        @PathVariable month: Int,
        authentication: Authentication
    ): ResponseEntity<List<Lesson>> {
        val journal = journalService.findOneById(id).getOrNull()
        val filteredLessons: List<Lesson> = journal?.lessons
            ?.filter { lesson -> lesson.date.monthValue == month }!!

        return ResponseEntity.ok(filteredLessons.map { lesson: Lesson ->
lesson.copy(marks = lesson.marks.filter { mark -> mark.student ==
authentication.principal as Int }) })
    }
}
```

--- Вміст файлу: ./djournal/controller/TestsController.kt ---

```
package com.djournal.controller
```

```
import com.djournal.domain.EPQTestData
import com.djournal.domain.RIASECTestData
import com.djournal.domain.dto.TestCompletionDTO
import com.djournal.service.EPQService
import com.djournal.service.RIASECService
import com.djournal.service.UserService
import org.springframework.http.ResponseEntity
import org.springframework.web.bind.annotation.GetMapping
import org.springframework.web.bind.annotation.PostMapping
import org.springframework.web.bind.annotation.RequestMapping
import org.springframework.web.bind.annotation.RestController
import java.security.Principal

@RestController
@RequestMapping("/tests")
class TestsController(
    val epqService: EPQService,
    val riasecService: RIASECService,
    val userService: UserService,
) {
    @GetMapping()
    fun getTestStates(userId: Principal) :
ResponseEntity<TestCompletionDTO> {
        val testCompletion = TestCompletionDTO(
            epq = epqService.findByUserId(userId.name.toInt()) != null,
            riasec = riasecService.findByUserId(userId.name.toInt()) != null
        )
        return ResponseEntity.ok(testCompletion)
    }

    @PostMapping("/save/riasec")
    fun riasecSave(testData: RIASECTestData, userId: Principal) :
ResponseEntity<Unit>
    {
        testData.user = userService.findById(userId.name.toInt())
        riasecService.save(testData)
        return ResponseEntity.ok().build()
    }
```

```
        @PostMapping("/save/epq")
        fun riasecSave(testData: EPQTestData, userId: Principal) :
ResponseEntity<Unit>
        {
            testData.user = userService.findById(userId.name.toInt())
            epqService.save(testData)
            return ResponseEntity.ok().build()
        }
    }
    --- Вміст файлу: ./djournal/controller/UserController.kt ---
    package com.djournal.controller

    import com.djournal.domain.User
    import com.djournal.domain.UserRole
    import com.djournal.service.UserService
    import org.springframework.http.ResponseEntity
    import org.springframework.web.bind.annotation.*

    @RestController
    @RequestMapping("/users")
    class UserController(
        private val userService: UserService,
    ) {
        @GetMapping
        fun getAllUsers() : ResponseEntity<List<User>>
        {
            return ResponseEntity.ok(userService.findAll())
        }
        @GetMapping("/teachers")
        fun getTeachers() : ResponseEntity<List<User>>
        {
            return
ResponseEntity.ok(userService.findByRole(UserRole.TEACHER))
        }
        @GetMapping("/students")
        fun getStudents() : ResponseEntity<List<User>>
        {
            return ResponseEntity.ok(userService.findByRole(UserRole.CLIENT))
```

```
    }
    @GetMapping("/admins")
    fun getAdmins() : ResponseEntity<List<User>>
    {
        return ResponseEntity.ok(userService.findByRole(UserRole.ADMIN))
    }

    @PostMapping("/save")
    fun register(@RequestBody user: User) : ResponseEntity<User> {
        return ResponseEntity.ok(userService.save(user))
    }
}
--- Вміст файлу: ./djournal/controller/test/TestPredictController.kt ---
package com.djournal.controller.test

import com.djournal.service.PredictionService
import org.springframework.http.ResponseEntity
import org.springframework.web.bind.annotation.PostMapping
import org.springframework.web.bind.annotation.RestController

@RestController
class TestPredictController(
    val predictionService: PredictionService
) {
    @PostMapping("/testPredict")
    fun predict() : ResponseEntity<Any> {
        val map = createMap(
            listOf("Civic education", "Ukrainian language", "Biology", "Foreign language", "History of Ukraine", "Physics", "Informatics", "p points of EPQ test", "e points of EPQ test", "n points of EPQ test", "l points of EPQ test", "r points of RIASEC test", "i points of RIASEC test", "a points of RIASEC test", "s points of RIASEC test", "e points of RIASEC test", "c points of RIASEC test", "Ukrainian literature", "Foreign literature", "Algebra", "Geometry", "World history", "Jurisprudence", "Economy", "Geography", "Chemistry", "Astronomy", "Art", "Labor training. Technologies", "Basics of health", "Physical culture", "Defense of Ukraine"),
            listOf(9, 7, 7, 6, 8, 10, 6, 4, 12, 0, 12, 1, 3, 8, 7, 7, 7, 11, 9, 11, 8, 12, 11, 6, 9, 8, 12, 6, 12, 11, 9, 10)
```

```
            )
            return ResponseEntity.ok(predictionService.predictTopN(map,5))
        }

        private fun createMap(keys: List<String>, values: List<Int>):
Map<String, Int> {
            val map = mutableMapOf<String, Int>()
            for (i in keys.indices) {
                map[keys[i]] = values[i]
            }
            return map
        }
    }
    --- Вміст файлу: ./djournal/domain/Classes.kt ---
    package com.djournal.domain

    import jakarta.persistence.*


    @Entity(name="classes")
    data class Classes(
        @Id
        @GeneratedValue(strategy = GenerationType.IDENTITY)
        var id: Int?,
        var name: String,
        @OneToMany(targetEntity = User::class, )
        @JoinTable(name="student_classes",
            joinColumns = [JoinColumn(name = "classes_id",
referencedColumnName = "id")],
            inverseJoinColumns = [JoinColumn(name = "student_id",
referencedColumnName = "id")]
        )
        var students: List<User>?
    )

    --- Вміст файлу: ./djournal/domain/EPQTestData.kt ---
    package com.djournal.domain
```

```
import com.fasterxml.jackson.annotation.JsonBackReference
import com.fasterxml.jackson.annotation.JsonProperty
import jakarta.persistence.*

@Entity(name = "epq_data")
data class EPQTestData(
    @Id @GeneratedValue(strategy = GenerationType.IDENTITY)
    val id: Int?,
    @JsonProperty("P")
    val pPoints: Int,
    @JsonProperty("E")
    val ePoints: Int,
    @JsonProperty("N")
    val nPoints: Int,
    @JsonProperty("L")
    val lPoints: Int,
    @OneToOne
    @JoinColumn(name = "user_id", nullable = false)
    @JsonBackReference
    var user: User,
)
--- Вміст файлу: ./djournal/domain/Journal.kt ---
package com.djournal.domain

import com.djournal.domain.dto.JournalDTO
import jakarta.persistence.*


@Entity
data class Journal(
    @Id @GeneratedValue(strategy = GenerationType.IDENTITY)
    var id: Int?,
    var name: String,
    @ManyToOne(targetEntity = Subject::class)
    @JoinColumn(name = "subject")
    var subject: Subject,
    @OneToMany
    @JoinColumn(name = "journal")
```

```
    var workTypes: List<WorkType>,
    @OneToMany(fetch = FetchType.LAZY, cascade =
[CascadeType.REMOVE])
    @JoinColumn(name = "journal")
    var lessons: List<Lesson>,
    @ManyToMany(fetch = FetchType.LAZY)
    @JoinTable(
      name = "teacher_journal",
      joinColumns = [JoinColumn(name = "journal")],
      inverseJoinColumns = [JoinColumn(name = "users")],
    )
    var teachers: List<User>,
    @ManyToMany(fetch = FetchType.LAZY)
    @JoinTable(
      name = "student_journal",
      joinColumns = [JoinColumn(name = "journal")],
      inverseJoinColumns = [JoinColumn(name = "users")]
    )
    var students: List<User>

) {
    fun toDTO(): JournalDTO {
      return JournalDTO(this)
    }
}

--- Вміст файлу: ./djournal/domain/Lesson.kt ---
package com.djournal.domain

import jakarta.persistence.*
import java.time.LocalDate

@Entity
data class Lesson(
    @Id @GeneratedValue(strategy = GenerationType.IDENTITY)
    val id: Int? = null,
    val date: LocalDate,
    val journal: Int,
```

```
        val lessonNumber: Int = 0,
        val theme: String,
        val homeWork: String,
        @OneToMany(mappedBy = "lesson", cascade =
[CascadeType.REMOVE], fetch = FetchType.LAZY)
        val marks: List<Mark> = emptyList(),
        @ManyToMany(fetch = FetchType.LAZY, cascade =
[CascadeType.PERSIST])
        @JoinColumn(name = "lesson_work_type", updatable = false)
        val workTypes: List<WorkType>? = emptyList(),
    )
    --- Вміст файлу: ./djournal/domain/LoginState.kt ---
    package com.djournal.domain

    import org.springframework.http.HttpStatus

    enum class LoginState {

        AUTHENTICATED {
            override fun getResponseStatus(): HttpStatus {
                return HttpStatus.OK
            }
        },

        NOT_AUTHENTICATED {
            override fun getResponseStatus(): HttpStatus {
                return HttpStatus.BAD_REQUEST
            }
        };

        abstract fun getResponseStatus(): HttpStatus
    }
    --- Вміст файлу: ./djournal/domain/Mark.kt ---
    package com.djournal.domain

    import com.fasterxml.jackson.annotation.JsonBackReference
    import jakarta.persistence.*
```

```
@Entity
data class Mark(
    @Id @GeneratedValue(strategy = GenerationType.IDENTITY)
    var id: Int? = null,
    var student: Int,
    @ManyToOne
    @JoinColumn(name = "lesson", nullable = false)
    @JsonBackReference
    var lesson: Lesson,
    var workType: Int,
    var mark: Double = 0.0,
)

--- Вміст файлу: ./djournal/domain/RIASECTestData.kt ---
package com.djournal.domain

import com.fasterxml.jackson.annotation.JsonBackReference
import com.fasterxml.jackson.annotation.JsonProperty
import jakarta.persistence.*

@Entity(name = "riasec_data")
data class RIASECTestData(
    @Id @GeneratedValue(strategy = GenerationType.IDENTITY)
    val id: Int?,
    @JsonProperty("R")
    val rPoints: Int,
    @JsonProperty("I")
    val iPoints: Int,
    @JsonProperty("A")
    val aPoints: Int,
    @JsonProperty("S")
    val sPoints: Int,
    @JsonProperty("E")
    val ePoints: Int,
    @JsonProperty("C")
    val cPoints: Int,
    @OneToOne
    @JoinColumn(name = "user_id", nullable = false)
```

```
    @JsonBackReference
    var user: User,
)
--- Вміст файлу: ./djournal/domain/Subject.kt ---
package com.djournal.domain

import jakarta.persistence.Entity
import jakarta.persistence.GeneratedValue
import jakarta.persistence.GenerationType
import jakarta.persistence.Id

@Entity
data class Subject(
    @Id @GeneratedValue(strategy = GenerationType.IDENTITY)
    var id: Int?,
    var name: String
)

--- Вміст файлу: ./djournal/domain/User.kt ---
package com.djournal.domain

import jakarta.persistence.*
import java.util.Date

@Entity
@Table(name = "users")
data class User(
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    var id: Int?,
    val userName: String,
    val password: String,
    @field:Enumerated(EnumType.STRING)
    var role: UserRole,
    var isActive: Boolean,
    var firstName: String,
    var lastName: String,
    var patronymic: String,
```

```
    @Temporal(TemporalType.DATE)
    var birthDate: Date
)

--- Вміст файлу: ./djournal/domain/UserRole.kt ---
package com.djournal.domain

enum class UserRole {
    OWNER,
    ADMIN,
    TEACHER,
    CLIENT
}

--- Вміст файлу: ./djournal/domain/UserSession.kt ---
package com.djournal.domain

import java.util.*

data class UserSession(
    var id: UUID,
    var user: User
)

--- Вміст файлу: ./djournal/domain/WorkType.kt ---
package com.djournal.domain

import jakarta.persistence.Entity
import jakarta.persistence.GeneratedValue
import jakarta.persistence.GenerationType
import jakarta.persistence.Id

@Entity
data class WorkType(
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    var id: Int?,
    var name: String,
```

```
    var maxCount: Int = 0,
    var maxCost: Double = 0.0,
    val journal: Int

)

--- Вміст файлу: ./djournal/domain/dto/JournalCreateDto.java ---
package com.djournal.domain.dto;

import com.djournal.domain.WorkType;

import java.util.List;

public class JournalCreateDto {
    String name;
    int subjectId;
    List<Integer> students;
    List<Integer> additionalTeachers;
    List<WorkType> workTypes;
}
//TODO: complete journal endpoint
//TODO: make dto for journal settings and other shit like data, etc.
//TODO: make endpoint to create journal

--- Вміст файлу: ./djournal/domain/dto/JournalDTO.kt ---
package com.djournal.domain.dto

import com.djournal.domain.Journal
import com.djournal.domain.Subject
import com.djournal.domain.WorkType

class JournalDTO(journal: Journal) {
    var id: Int = journal.id!!
    var name: String = journal.name
    var subject: Subject? = journal.subject
    var workTypes: List<WorkType> = journal.workTypes
    var teachers: List<UserDTO> = journal.teachers.map { UserDTO(it) }
    var students: List<UserDTO> = journal.students.map { UserDTO(it) }
```

```
}
--- Вміст файлу: ./djournal/domain/dto/JournalRenderDto.java ---
package com.djournal.domain.dto;

import com.djournal.domain.Journal;
import com.djournal.domain.Subject;
import com.djournal.domain.WorkType;
import lombok.Value;

import java.util.List;

@Value
public class JournalRenderDto {
    Integer id;
    String name;
    Subject subject;
    List<UserDTO> teachers;
    List<UserDTO> students;
    List<WorkType> workTypes;

    public JournalRenderDto(Journal journal,
                            List<UserDTO> teachers,
                            List<UserDTO> students) {
        this.id = journal.getId();
        this.name = journal.getName();
        this.subject = journal.getSubject();
        this.teachers = teachers;
        this.students = students;
        this.workTypes = journal.getWorkTypes();
    }
}

--- Вміст файлу: ./djournal/domain/dto/LessonEditDTO.kt ---
package com.djournal.domain.dto

import java.time.LocalDate

data class LessonEditDTO(
```

```
    val id: Int?,
    val date: LocalDate,
    val journal: Int,
    val lessonNumber: Int,
    val theme: String,
    val homeWork: String,
    val workTypes: List<Int>
)

--- Вміст файлу: ./djournal/domain/dto/LessonMarksDto.kt ---
package com.djournal.domain.dto

class LessonMarksDto(
    var lesson: Int,
    var student: Int,
    var marks: Map<Int, Double>
)

--- Вміст файлу: ./djournal/domain/dto/LoginDTO.kt ---
package com.djournal.domain.dto

data class LoginDTO(
    val username: String,
    val password: String
)
--- Вміст файлу: ./djournal/domain/dto/LoginPasswordDto.java ---
package com.djournal.domain.dto;

import lombok.Value;

@Value
public class LoginPasswordDto {
    String login;
    String password;
}

--- Вміст файлу: ./djournal/domain/dto/Student.kt ---
package com.djournal.domain.dto
```

```
import com.djournal.domain.User

data class Student(
    val user: User,
    val schoolClass: String
) {
    var id: Int? = user.id
    var firstName: String = user.firstName
    var lastName: String = user.lastName
    var className: String = schoolClass
}

--- Вміст файлу: ./djournal/domain/dto/TestCompletionDTO.kt ---
package com.djournal.domain.dto

data class TestCompletionDTO(
    val epq: Boolean,
    val riasec: Boolean
) {
}
--- Вміст файлу: ./djournal/domain/dto/TokensDto.java ---
package com.djournal.domain.dto;

import lombok.Value;

@Value
public class TokensDto {
    String accessToken;
    String refreshToken;
}

--- Вміст файлу: ./djournal/domain/dto/UserDTO.kt ---
package com.djournal.domain.dto

import com.djournal.domain.User
```

```
class UserDTO(user: User) {
    var id: Int? = user.id
    var firstName: String = user.firstName
    var lastName: String = user.lastName
}

--- Вміст файлу: ./djournal/domain/dto/UserTokensDTO.kt ---
package com.djournal.domain.dto

data class UserTokensDTO(
    val accessToken: String,
    val refreshToken: String
)

--- Вміст файлу: ./djournal/exception/AuthenticationException.java ---
package com.djournal.exception;

public class AuthenticationException extends RuntimeException {
    public AuthenticationException() {
        super();
    }

    public AuthenticationException(String message) {
        super(message);
    }
}

--- Вміст файлу: ./djournal/exception/DataNotFoundException.kt ---
package com.djournal.exception

class DataNotFoundException : RuntimeException()
--- Вміст файлу: ./djournal/exception/InvalidOperationException.java ---
package com.djournal.exception;

public class InvalidOperationException extends RuntimeException {
    public InvalidOperationException() {
        super();
    }
```

```
    public InvalidOperationException(String message) {
        super(message);
    }
}

--- Вміст файлу: ./djournal/exception/UserNotFoundException.kt ---
package com.djournal.exception

class UserNotFoundException : RuntimeException()
--- Вміст файлу: ./djournal/repository/ClassesRepository.kt ---
package com.djournal.repository

import com.djournal.domain.Classes
import org.springframework.data.jpa.repository.JpaRepository
import org.springframework.stereotype.Repository

@Repository
interface ClassesRepository : JpaRepository<Classes, Int>

--- Вміст файлу: ./djournal/repository/EPQRepository.kt ---
package com.djournal.repository

import com.djournal.domain.EPQTestData
import org.springframework.data.jpa.repository.JpaRepository

interface EPQRepository : JpaRepository<EPQTestData, Int>{
    fun findByUserId(userId: Int): EPQTestData?
}
--- Вміст файлу: ./djournal/repository/JournalRepository.kt ---
package com.djournal.repository

import com.djournal.domain.Journal
import org.springframework.data.jpa.repository.EntityGraph
import org.springframework.data.jpa.repository.JpaRepository
import org.springframework.data.jpa.repository.Query
import org.springframework.data.repository.query.Param
import org.springframework.stereotype.Repository
```

```
import java.util.*


@Repository
interface JournalRepository : JpaRepository<Journal, Int> {
    fun findJournalsByStudents_Id(id: Int): List<Journal>
    fun findJournalsByTeachers_Id(id: Int): List<Journal>

    @Query(
        """
    SELECT j FROM Journal j
    LEFT JOIN FETCH j.lessons l
    LEFT JOIN FETCH l.marks
    WHERE j.id = :id
    """
    )
    fun findByIdWithLessonsAndMarks(@Param("id") id: Int):
Optional<Journal>

    @EntityGraph(attributePaths = ["lessons", "lessons.marks"])
    @Query("SELECT j FROM Journal j WHERE j.id = :id")
    fun findByIdWithLessonsAndMarks1(@Param("id") id: Int): Journal?
}

--- Вміст файлу: ./djournal/repository/LessonRepository.kt ---
package com.djournal.repository

import com.djournal.domain.Lesson
import org.springframework.data.jpa.repository.JpaRepository
import org.springframework.stereotype.Repository

@Repository
interface LessonRepository : JpaRepository<Lesson, Int>

--- Вміст файлу: ./djournal/repository/MarkRepository.kt ---
package com.djournal.repository

import com.djournal.domain.Mark
```

```
import org.springframework.data.jpa.repository.JpaRepository
import org.springframework.stereotype.Repository

@Repository
interface MarkRepository : JpaRepository<Mark, Int> {
    fun findByLesson_IdAndStudentAndWorkType(lesson: Int, student: Int,
workType: Int): Mark?
}

--- Вміст файлу: ./djournal/repository/RIASECRepository.kt ---
package com.djournal.repository

import com.djournal.domain.RIASECTestData
import org.springframework.data.jpa.repository.JpaRepository

interface RIASECRepository : JpaRepository<RIASECTestData, Int> {
    fun findByUserId(userId: Int): RIASECTestData?
}
--- Вміст файлу: ./djournal/repository/SubjectRepository.kt ---
package com.djournal.repository

import com.djournal.domain.Subject
import org.springframework.data.jpa.repository.JpaRepository
import org.springframework.stereotype.Repository

@Repository
interface SubjectRepository : JpaRepository<Subject, Int>
--- Вміст файлу: ./djournal/repository/UserRepository.kt ---
package com.djournal.repository

import com.djournal.domain.User
import com.djournal.domain.UserRole
import org.springframework.data.jpa.repository.JpaRepository
import org.springframework.stereotype.Repository

@Repository
interface UserRepository : JpaRepository<User, Int> {
```

```
    fun findByUserNameAndPassword(username: String, password: String):
User?
    fun findByRole(role: UserRole): MutableList<User>
}

--- Вміст файлу: ./djournal/repository/WorkTypeRepository.kt ---
package com.djournal.repository

import org.springframework.data.jpa.repository.JpaRepository
import org.springframework.stereotype.Repository
import com.djournal.domain.WorkType as WorkTypeClass

@Repository
interface WorkTypeRepository : JpaRepository<WorkTypeClass, Int>

--- Вміст файлу: ./djournal/security/JWTFilter.kt ---
package com.djournal.security

import com.djournal.domain.LoginState
import com.djournal.domain.dto.UserTokensDTO
import com.djournal.exception.UserNotFoundException
import com.djournal.service.LoginService
import com.djournal.util.HTTPUtil
import jakarta.servlet.FilterChain
import jakarta.servlet.http.HttpServletRequest
import jakarta.servlet.http.HttpServletResponse
import
org.springframework.security.authentication.UsernamePasswordAuthenticationTo
ken
import org.springframework.security.core.GrantedAuthority
import org.springframework.security.core.context.SecurityContextHolder
import org.springframework.stereotype.Component
import org.springframework.web.filter.OncePerRequestFilter

@Component
class JWTFilter(
    val loginService: LoginService,
) : OncePerRequestFilter() {
```

```
        override fun doFilterInternal(
            request: HttpServletRequest,
            response: HttpServletResponse,
            filterChain: FilterChain
        ) {
            println("Request to ${request.method} ${request.requestURI}{}")
            var authenticated = false
            var token = request.cookies?.firstOrNull { cookie -> cookie.name ==
"token" }?.value

            if (loginService.loginByToken(token) ==
LoginState.AUTHENTICATED)
            {
                authenticated = true
            }
            else
            {
                val refreshToken = request.cookies?.firstOrNull { cookie ->
cookie.name == "refresh-token" }?.value
                val newTokens = loginService.refreshTokens(refreshToken)
                if (newTokens != null)
                {
                    HTTPUtil.applyNewTokens(response, newTokens)
                    token = newTokens.accessToken
                    authenticated = true
                }
            }
            if (authenticated) {
                val user = token.let { loginService.getUserFromToken(it!!) }
                if (user != null) {
                    val context = SecurityContextHolder.getContext()
                    val authenticationToken =
                        UsernamePasswordAuthenticationToken.authenticated(user.id,
user, mutableListOf(GrantedAuthority {"ROLE_"+user.role.name}))
                    context.authentication = authenticationToken
                }
            }
            else {
```

```
                HTTPUtil.suspendCookie("role", false)
                HTTPUtil.suspendCookie("token", true)
                HTTPUtil.suspendCookie("refresh-token", true)
            }
            response.addCookie(HTTPUtil.createPublicCookie("Authenticated",
authenticated.toString()))
            filterChain.doFilter(request, response)
        }

    }

    --- Вміст файлу: ./djournal/security/JWTProvider.kt ---
    package com.djournal.security

    import com.djournal.domain.User
    import com.djournal.domain.dto.UserTokensDTO
    import com.djournal.service.UserService
    import io.jsonwebtoken.Claims
    import io.jsonwebtoken.ExpiredJwtException
    import io.jsonwebtoken.JwtParser
    import io.jsonwebtoken.Jwts
    import io.jsonwebtoken.io.Decoders
    import io.jsonwebtoken.security.Keys
    import org.springframework.beans.factory.annotation.Value
    import org.springframework.stereotype.Component
    import java.util.*
    import java.util.concurrent.TimeUnit

    @Component
    class JWTProvider(
        val userService: UserService,
        @Value("\${jwt.secret}") final val secretKey: String,
        @Value("\${jwt.expiration-time-in-minutes.access}") val
accessTokenExpirationTime: Long? = null,
        @Value("\${jwt.expiration-time-in-minutes.refresh}") val
refreshTokenExpirationTime: Long? = null,
        @Value("\${jwt.prefix}") val tokenPrefix: String? = null
    ) {
```

```
        private var parser: JwtParser =

Jwts.parser().verifyWith(Keys.hmacShaKeyFor(Decoders.BASE64.decode(secret
Key))).build()

        fun generateTokens(user: User): UserTokensDTO {
            val clearUser = user.copy(password = "")
            return UserTokensDTO(
                accessToken = (generateToken(clearUser, false)),
                refreshToken = (generateToken(clearUser, true))
            )
        }

        private fun generateToken(user: User, isRefreshToken: Boolean): String {
            return Jwts.builder()
                .claim("id", user.id)
                .claims()
                .subject(user.userName)
                .and()
                .issuedAt(Date(System.currentTimeMillis()))
                .expiration(Date(Date().time + TimeUnit.MINUTES.toMillis((if
(isRefreshToken) refreshTokenExpirationTime else
accessTokenExpirationTime)!!)))

.signWith(Keys.hmacShaKeyFor(Decoders.BASE64.decode(secretKey)))
                .compact()
        }

        fun isTokenValid(token: String): Boolean {
            try {
                val claims = extractAllClaims(token)
                return claims.expiration.after(Date(System.currentTimeMillis()))
            } catch (e: ExpiredJwtException) {
                return false
            }
        }
```

```
            fun getUser(token: String): User? {
                val userIdClaim = extractAllClaims(token)["id"]
                val userId = userIdClaim as Int
                return userService.findById(userId)
            }

            private fun extractAllClaims(token: String): Claims {
                return parser.parseSignedClaims(token.replace(tokenPrefix!!,
"")).payload
            }
        }
        --- Вміст файлу: ./djournal/security/SecurityConfiguration.kt ---
        package com.djournal.security

        import com.djournal.domain.UserRole
        import org.springframework.context.annotation.Bean
        import org.springframework.context.annotation.Configuration
        import org.springframework.security.authentication.AuthenticationManager
        import
org.springframework.security.authentication.AuthenticationServiceException
        import
org.springframework.security.config.annotation.web.builders.HttpSecurity
        import
org.springframework.security.config.annotation.web.configuration.EnableWebSec
urity
        import org.springframework.security.config.annotation.web.invoke
        import org.springframework.security.core.Authentication
        import org.springframework.security.web.SecurityFilterChain
        import
org.springframework.security.web.authentication.www.BasicAuthenticationFilter


        @Configuration
        @EnableWebSecurity
        class SecurityConfiguration {

            @Bean
```

```
        fun securityWebFilterChain(http: HttpSecurity, jwtFilter: JWTFilter):
SecurityFilterChain {
            http.invoke {
                authorizeRequests {
                    authorize("/", authenticated)
                    authorize("/login/**", permitAll)
                    authorize("/logout", permitAll)
                    authorize("/journal/**", authenticated)
                    authorize("/journal/all/student", hasRole("CLIENT"))
                    authorize("/student/**", hasRole("CLIENT"))
                    authorize("/lesson/**", authenticated)
                    authorize("/mark/**", authenticated)
                    authorize("/testPredict", authenticated)
                    authorize("/tests/**", authenticated)
                    authorize("/prediction", authenticated)
                    authorize("/classes", hasRole(UserRole.ADMIN.toString()))
                    authorize("/users/**", hasAnyRole(UserRole.ADMIN.toString(),
UserRole.OWNER.toString()))
                    authorize(anyRequest, denyAll)
                }
                csrf { disable() }
                cors { disable() }
                addFilterAfter<BasicAuthenticationFilter>(jwtFilter)
            }

            return http.build()

        }

        @Bean
        fun noopAuthenticationManager(): AuthenticationManager {
            return AuthenticationManager { _: Authentication? ->
                throw AuthenticationServiceException("Authentication is disabled")
            }
        }
    }
    --- Вміст файлу: ./djournal/service/ClassesService.kt ---
    package com.djournal.service
```

```
import com.djournal.domain.Classes
import com.djournal.repository.ClassesRepository
import org.springframework.stereotype.Service

@Service
class ClassesService(
    private val classesRepository: ClassesRepository
) {
    fun getClasses(): List<Classes> {
        return classesRepository.findAll()
    }

    fun saveClass(classData: Classes) {
        classesRepository.save(classData)
    }
}
--- Вміст файлу: ./djournal/service/EPQService.kt ---
package com.djournal.service

import com.djournal.domain.EPQTestData

interface EPQService {
    fun save(testData: EPQTestData) : EPQTestData
    fun update(testData: EPQTestData)
    fun findById(id: Int): EPQTestData
    fun findByUserId(userId: Int): EPQTestData?
}
--- Вміст файлу: ./djournal/service/JournalService.kt ---
package com.djournal.service

import com.djournal.domain.Journal
import com.djournal.domain.User
import java.util.*

interface JournalService {
    //get empty list of journals that student mentioned in
    fun findAllByStudent(user: User): List<Journal>
```

```
    //get empty journals for list on teachers page etc.
    fun findAllByTeacherId(id: Int): List<Journal>
    fun findAllByStudentsId(id: Int): List<Journal>
    fun findOneById(id: Int): Optional<Journal>
}
--- Вміст файлу: ./djournal/service/LessonService.kt ---
package com.djournal.service

import com.djournal.domain.Lesson

interface LessonService {
    fun findById(lessonId: Int): Lesson?
    fun save(lesson: Lesson): Lesson
    fun delete(lesson: Lesson)
    fun deleteById(id: Int)
}
--- Вміст файлу: ./djournal/service/LoginService.kt ---
package com.djournal.service

import com.djournal.domain.LoginState
import com.djournal.domain.User
import com.djournal.domain.dto.UserTokensDTO
import jakarta.servlet.http.HttpServletResponse

interface LoginService {
    fun loginByUsernameAndPassword(response: HttpServletResponse,
username: String, password: String): LoginState
    fun registerUser(user: User): User
    fun loginByToken(token: String?): LoginState
    fun refreshTokens(refreshToken: String?): UserTokensDTO?
    fun getUserFromToken(token: String): User?
}
--- Вміст файлу: ./djournal/service/MarkService.kt ---
package com.djournal.service

import com.djournal.domain.dto.LessonMarksDto
import jakarta.transaction.Transactional
```

```
interface MarkService {
    @Transactional
    fun saveMarksByLesson(lessonMarks: LessonMarksDto)
}
--- Вміст файлу: ./djournal/service/PredictionService.kt ---
package com.djournal.service

import com.djournal.service.impl.WorkTypeService
import hex.genmodel.easy.EasyPredictModelWrapper
import hex.genmodel.easy.RowData
import hex.genmodel.easy.prediction.MultinomialModelPrediction
import jakarta.transaction.Transactional
import org.springframework.stereotype.Service
import java.util.stream.Collectors

@Service
class PredictionService(
    private val predictModelWrapper: EasyPredictModelWrapper,
    private val journalService: JournalService,
    private val userService: UserService,
    private val workTypeService: WorkTypeService,
    private val epqService: EPQService,
    private val riasecService: RIASECService
) {

    @Throws(Exception::class)
    fun predictTopN(input: Map<String, Any>, topN: Int):
MutableList<String>? {
        // Конвертуємо дані у формат RowData
        val row = RowData()
        input.forEach { (key: String?, value: Any?) -> row[key] =
value.toString() }

        // Отримуємо передбачення
        val prediction: MultinomialModelPrediction =
predictModelWrapper.predictMultinomial(row)
```

```
            // Сортуємо класи за ймовірністю
            val probabilities: MutableMap<String, Double> = HashMap()
            for (i in prediction.classProbabilities.indices) {
                probabilities[predictModelWrapper.responseDomainValues[i]] =
prediction.classProbabilities[i]
            }

            return probabilities.entries
                .stream()
                .sorted { e1: Map.Entry<String, Double>, e2: Map.Entry<String?,
Double?> ->
                    e2.value!!.compareTo(e1.value)
                }
                .map { data -> data.key }
                .limit(topN.toLong())
                .collect(Collectors.toUnmodifiableList())
        }

        @Transactional
        fun predictJobs(userId: Int): MutableList<String>? {
            val userData = mutableMapOf<String, Int>()
            val user = userService.findById(userId)
            val journals = journalService.findAllByStudent(user)
            for (journal in journals) {
                val subject = journal.subject
                var studentMarks = 0.0
                var maxMarks = 0.0
                for (lesson in journal.lessons) {
                    val marks = lesson.marks
                    for (mark in marks)
                    {
                        studentMarks += mark.mark
                        maxMarks +=
workTypeService.findWorkTypeById(mark.workType).maxCost
                    }
                }
                userData[subject.name] = (studentMarks / maxMarks * 100).toInt()
```

```
        }
        val riasecTestData = riasecService.findByUserId(userId)
        if (riasecTestData != null)
        {
            userData["r points of RIASEC test"] = riasecTestData.rPoints
            userData["i points of RIASEC test"] = riasecTestData.iPoints
            userData["a points of RIASEC test"] = riasecTestData.aPoints
            userData["s points of RIASEC test"] = riasecTestData.sPoints
            userData["e points of RIASEC test"] = riasecTestData.ePoints
            userData["c points of RIASEC test"] = riasecTestData.cPoints
        }
        val epqTestData = epqService.findByUserId(userId)
        if (epqTestData != null)
        {
            userData["p points of EPQ test"] = epqTestData.pPoints
            userData["e points of EPQ test"] = epqTestData.ePoints
            userData["n points of EPQ test"] = epqTestData.nPoints
            userData["l points of EPQ test"] = epqTestData.lPoints

        }
        println(userData)
        return predictTopN(userData, 5)
    }
}

--- Вміст файлу: ./djournal/service/RIASECService.kt ---
package com.djournal.service

import com.djournal.domain.RIASECTestData

interface RIASECService {
    fun save(testData: RIASECTestData) : RIASECTestData
    fun update(testData: RIASECTestData)
    fun findById(id: Int): RIASECTestData
    fun findByUserId(userId: Int): RIASECTestData?
}
--- Вміст файлу: ./djournal/service/UserService.kt ---
package com.djournal.service
```

```
import com.djournal.domain.User
import com.djournal.domain.UserRole

interface UserService {
    fun findAll(): List<User>
    fun findById(id: Int): User
    fun save(user: User): User
    fun delete(id: Int)
    fun findByUsernameAndPassword(username: String, password: String):
User?
    fun findByRole(role: UserRole): List<User>
}
--- Вміст файлу: ./djournal/service/authentication/LoginServiceImpl.kt ---
package com.djournal.service.authentication

import com.djournal.domain.LoginState
import com.djournal.domain.User
import com.djournal.domain.dto.UserTokensDTO
import com.djournal.exception.UserNotFoundException
import com.djournal.security.JWTProvider
import com.djournal.service.LoginService
import com.djournal.service.UserService
import com.djournal.util.HTTPUtil
import jakarta.servlet.http.HttpServletResponse
import org.springframework.stereotype.Service

@Service
class LoginServiceImpl(
    var jwtProvider: JWTProvider,
    var userService: UserService,
) : LoginService {
    override fun loginByUsernameAndPassword(
        response: HttpServletResponse,
        username: String,
        password: String
    ): LoginState {
        val user =
```

```
                userService.findByUsernameAndPassword(username, password) ?:
return LoginState.NOT_AUTHENTICATED

            val userTokens = jwtProvider.generateTokens(user)
            HTTPUtil.applyNewTokens(response, userTokens)
            response.addCookie(HTTPUtil.createPublicCookie("Authenticated",
"True"))
            response.addCookie(HTTPUtil.createPublicCookie("role",
user.role.name))
            return LoginState.AUTHENTICATED
        }

        override fun registerUser(user: User): User {
            val registeredUser = userService.save(user)
            return registeredUser
        }

        override fun loginByToken(token: String?): LoginState {
            if (token == null || !jwtProvider.isTokenValid(token)) {
                return LoginState.NOT_AUTHENTICATED
            }
                return LoginState.AUTHENTICATED
        }

        override fun refreshTokens(refreshToken: String?): UserTokensDTO? {
            if (refreshToken != null && jwtProvider.isTokenValid(refreshToken))
{
                try {
                    val user = jwtProvider.getUser(refreshToken)?.copy(password =
"") ?: throw UserNotFoundException()
                    val newTokens: UserTokensDTO =
jwtProvider.generateTokens(user)
                    return newTokens
                } catch (_: UserNotFoundException) {}
            }
            return null
        }
```

```
    override fun getUserFromToken(token: String): User? {
        return jwtProvider.getUser(token)
    }
}

--- Вміст файлу: ./djournal/service/impl/EPQServiceImpl.kt ---
package com.djournal.service.impl

import com.djournal.domain.EPQTestData
import com.djournal.repository.EPQRepository
import com.djournal.service.EPQService
import org.springframework.stereotype.Service

@Service
class EPQServiceImpl(
    val repository: EPQRepository
) : EPQService {
    override fun save(testData: EPQTestData): EPQTestData {
        return repository.save(testData)
    }

    override fun update(testData: EPQTestData) {
        repository.save(testData)
    }

    override fun findById(id: Int): EPQTestData {
        return repository.findById(id).orElse(null)
    }

    override fun findByUserId(userId: Int): EPQTestData? {
        return repository.findByUserId(userId)
    }
}
--- Вміст файлу: ./djournal/service/impl/JournalServiceImpl.kt ---
package com.djournal.service.impl

import com.djournal.domain.Journal
import com.djournal.domain.User
```

```
import com.djournal.exception.DataNotFoundException
import com.djournal.repository.JournalRepository
import com.djournal.service.JournalService
import com.djournal.service.UserService
import org.springframework.stereotype.Service
import java.util.*

@Service
class JournalServiceImpl(
    private val repository: JournalRepository,
    private val userService: UserService,
    private val subjectService: SubjectService,
) : JournalService {

    //get empty list of journals that student mentioned in
    override fun findAllByStudent(user: User): List<Journal> {
        return repository.findJournalsByStudents_Id(user.id!!)
    }

    //get empty journals for list on teachers page etc.
    override fun findAllByTeacherId(id: Int): List<Journal> {
        return repository.findJournalsByTeachers_Id(id)
    }

    override fun findAllByStudentsId(id: Int): List<Journal> {
        return repository.findJournalsByStudents_Id(id)
    }

    override fun findOneById(id: Int): Optional<Journal> {
        return repository.findById(id) ?: throw DataNotFoundException()
    }


}

--- Вміст файлу: ./djournal/service/impl/LessonServiceImpl.kt ---
package com.djournal.service.impl
```

```
import com.djournal.domain.Lesson
import com.djournal.exception.DataNotFoundException
import com.djournal.repository.LessonRepository
import com.djournal.service.LessonService
import org.springframework.stereotype.Service
import kotlin.jvm.optionals.getOrNull

@Service
class LessonServiceImpl(
    private val repository: LessonRepository,
) : LessonService {

    override fun findById(lessonId: Int): Lesson? {
        return repository.findById(lessonId).getOrNull() ?: throw
DataNotFoundException()
    }

    override fun save(lesson: Lesson): Lesson {
        return repository.save(lesson)
    }

    override fun delete(lesson: Lesson) {
        repository.delete(lesson)
    }

    override fun deleteById(id: Int) {
        repository.deleteById(id)
    }
}

--- Вміст файлу: ./djournal/service/impl/MarkServiceImpl.kt ---
package com.djournal.service.impl

import com.djournal.domain.Mark
import com.djournal.domain.dto.LessonMarksDto
import com.djournal.repository.MarkRepository
import com.djournal.service.LessonService
import com.djournal.service.MarkService
```

```
import jakarta.transaction.Transactional
import org.springframework.stereotype.Service

@Service
class MarkServiceImpl(
    private val repository: MarkRepository,
    private val lessonService: LessonService
) : MarkService {
    @Transactional
    override fun saveMarksByLesson(lessonMarks: LessonMarksDto) {
        lessonMarks
            .marks
            .entries
            .map { (key, value) ->

repository.findByLesson_IdAndStudentAndWorkType(lessonMarks.lesson,
lessonMarks.student, key)
                    ?.copy(mark = value)
                    ?: Mark(
                        null,
                        lessonMarks.student,
                        lessonService.findById(lessonMarks.lesson)!!,
                        key,
                        value
                    )
            }
            .toMutableList()
            .forEach { mark ->
                repository.save(mark)

            }

    }
}

--- Вміст файлу: ./djournal/service/impl/RIASECServiceImpl.kt ---
package com.djournal.service.impl
```

```
import com.djournal.domain.RIASECTestData
import com.djournal.repository.RIASECRepository
import com.djournal.service.RIASECService
import org.springframework.stereotype.Service

@Service
class RIASECServiceImpl(
    val repository: RIASECRepository
) : RIASECService {
    override fun save(testData: RIASECTestData): RIASECTestData {
        return repository.save(testData)
    }

    override fun update(testData: RIASECTestData) {
        repository.save(testData)
    }

    override fun findById(id: Int): RIASECTestData {
        return repository.findById(id).orElse(null)
    }

    override fun findByUserId(userId: Int): RIASECTestData? {
        return repository.findByUserId(userId)
    }
}
--- Вміст файлу: ./djournal/service/impl/StudentService.kt ---
package com.djournal.service.impl

import org.springframework.stereotype.Service

@Service
class StudentService {
    fun getStudentMarks(StudentId: Int, offset: Int) {}
}

--- Вміст файлу: ./djournal/service/impl/SubjectService.kt ---
package com.djournal.service.impl
```

```
import com.djournal.repository.SubjectRepository
import org.springframework.stereotype.Service

@Service
class SubjectService(
    val subjectRepository: SubjectRepository
)

--- Вміст файлу: ./djournal/service/impl/UserServiceImpl.kt ---
package com.djournal.service.impl

import com.djournal.domain.User
import com.djournal.domain.UserRole
import com.djournal.repository.UserRepository
import com.djournal.service.UserService
import org.springframework.http.HttpStatus
import org.springframework.stereotype.Repository
import org.springframework.web.server.ResponseStatusException
import com.ibm.icu.text.Transliterator
import java.util.*

@Repository
class UserServiceImpl(
    private val repository: UserRepository
) : UserService {
    override fun findAll(): List<User> {
        return repository.findAll()
    }

    override fun findById(id: Int): User {
        return repository.findById(id)
            .orElseThrow { ResponseStatusException(HttpStatus.NOT_FOUND,
"Cannot find user with id $id") }
    }

    override fun save(user: User): User {
        if (user.id == null) {
            var newUser = user.copy(
```

```
                userName = generateUniqueLogin(user.lastName),
                password = generatePassword(),
            )

        }
        return repository.save(copyWithPasswordEncoded(user))
            ?: throw
ResponseStatusException(HttpStatus.INTERNAL_SERVER_ERROR, "Cannot
save user to database")
    }

    override fun delete(id: Int) {
        return repository.deleteById(id)
    }

    override fun findByUsernameAndPassword(username: String, password:
String): User? {
        return repository.findByUserNameAndPassword(username, password)
    }

    override fun findByRole(role: UserRole): List<User> {
        return repository.findByRole(role)
    }

    private fun copyWithPasswordEncoded(user: User): User {
        return user.copy(
            password = user.password,
            //TODO
            //encoder.encode(user.findPassword()),
        )
    }

    private fun generateUniqueLogin(name: String): String {
        val transliterator = Transliterator.getInstance("Cyrillic-Latin")
        val transliterated = transliterator.transliterate(name)
        val uniqueId = UUID.randomUUID().toString().substring(0, 4) //
Короткий унікальний ідентифікатор
        return "${transliterated.substring(0,4)}${uniqueId}"
```

```
    }

    private fun generatePassword() : String
    {
        return UUID.randomUUID().toString().substring(0,8)
    }

}

--- Вміст файлу: ./djournal/service/impl/WorkTypeService.kt ---
package com.djournal.service.impl

import com.djournal.domain.WorkType
import com.djournal.repository.WorkTypeRepository
import org.springframework.stereotype.Service

@Service
class WorkTypeService(
    private val repository: WorkTypeRepository
)
{
    fun findWorkTypeById(workTypeId: Int): WorkType
    {
        return
repository.findById(workTypeId).orElseThrow{RuntimeException("Worktype
with id ${workTypeId} not found!")}
    }
}


--- Вміст файлу: ./djournal/util/GenericSerializationUtil.kt ---
package com.djournal.util

import com.fasterxml.jackson.databind.ObjectMapper
import org.springframework.stereotype.Component

@Component
object GenericSerializationUtil {
```

```
    val objectMapper = ObjectMapper()

    fun <T> serialize(t: T): ByteArray {
        return objectMapper.writeValueAsBytes(t)
    }

    inline fun <reified T> deserialize(bytes: ByteArray): T {
        return objectMapper.readValue(bytes, T::class.java)
    }
}

--- Вміст файлу: ./djournal/util/HTTPUtil.kt ---
package com.djournal.util

import com.djournal.domain.dto.UserTokensDTO
import jakarta.servlet.http.Cookie
import jakarta.servlet.http.HttpServletResponse

class HTTPUtil {
    companion object {
        fun createSecuredCookie(name: String, value: String): Cookie {
            return Cookie(name, value).apply {
                isHttpOnly = true
                maxAge = -1
                path = "/"
            }
        }

        fun suspendCookie(name: String, secured: Boolean = true): Cookie {
            return Cookie(name, "null").apply {
                path = "/"
                isHttpOnly = secured
                maxAge = 0
            }
        }

        fun createPublicCookie(name: String, value: String): Cookie {
            return Cookie(name, value).apply {
```

```
                maxAge = -1
                path = "/"
            }
        }

        fun applyNewTokens(response: HttpServletResponse, userTokens:
UserTokensDTO) {
            response.addCookie(createSecuredCookie("token",
userTokens.accessToken))
            response.addCookie(createSecuredCookie("refresh-token",
userTokens.refreshToken))
        }

    }
}
```

ДЕКЛАРАЦІЯ

про дотримання академічної доброчесності

Я, _____

_____

_____

_____

Повністю вказується ПІБ та статус (освітня (освітньо-наукова) програма – для здобувачів вищої освіти, назва кваліфікаційної роботи)

що нижче підписалась/підписався, розуміючи та підтримуючи загальновизнані засади справедливості, доброчесності та законності,

ЗОБОВ'ЯЗУЮСЬ:

дотримуватися принципів та правил академічної доброчесності, що визначені законодавством України, локальними нормативними актами Донецького національного університету імені Василя Стуса, положеннями, правилами, умовами, визначеними іншими суб'єктами, та не допускати їх порушення.

ПІДТВЕРДЖУЮ:

що мені відомі положення статті 42 Закону України «Про освіту»;

що у даній роботі не представляла/представляв чиїсь роботи повністю або частково як свої власні. Там, де я скористалася/скористався працею інших, я зробила/зробив відповідні посилання на джерела інформації;

що дана робота не передавалась іншим особам і подається вперше, не порушує авторських та суміжних прав закріплених статтями 21-25 Закону України «Про авторське право та суміжні права», а дані та інформація не отримувались в недозволений спосіб.

УСВІДОМЛЮЮ:

що ця робота може бути перевірена університетом на плагіат або інші порушення академічної доброчесності, в тому числі з використанням спеціалізованих сервісів;

що у разі порушення академічної доброчесності, до мене можуть бути застосовані процедури, передбачені законодавством України та Кодексом академічної доброчесності та корпоративної етики Донецького національного університету імені Василя Стуса, іншими локальними нормативними актами університету, та я можу бути притягнута/притягнутий до академічної відповідальності.

_____ _____

(дата) (підпис)