Міністерство освіти і науки України

Національний університет «Києво-Могилянська академія»

Факультет інформатики

Кафедра мультимедійних систем

Кваліфікаційна робота

освітній ступінь – бакалавр

на тему: «**ВДОСКОНАЛЕННЯ СИСТЕМИ ЕЛЕКТРОННОГО НАВЧАННЯ**»

Виконав: студент 4-го року навчання,

Освітньої програми «Прикладна математика», 113

Тихий Роман Вікторович

Керівник Бублик В. В.,_____

Кандидат фіз.-мат. наук, доцент

Рецензент Глибовець А. М.

Кваліфікаційна робота захищена з оцінкою

_____

Секретар ЕК

_____

«____» _____

20____ р.

Київ – 20___

ЗМІСТ

ВСТУП

Передача і розповсюдження знань складають важливу ділянку суспільної діяльності і творчості. З деякого часу досягнення технологій уможливили постановку питання про вдосконалення методів і засобів освіти. В роботі [1] дано аналітичний огляд дистанційної освіти з її переходом до електронних систем. Там же розглянуті деякі розповсюджені системи управління навчальним процесом і відзначені особливості застосування і напрямки можливого розвитку системи MOODLE (Modular Object-Oriented Dynamic Learning Environment) стосовно досвіду, набутому на факультеті інформатики..

Дійсно, багато існуючих систем електронного навчання не надають можливості для командної роботи та групових проектів в повній мірі, а також не пристосовані до всіх можливих форматів навчання. Це може стати перешкодою як для студентів, які навчаються разом та мають спільні завдання, так і для викладачів, які ці завдання дають. Але завдяки відкритості та модульності систем управління навчальними процесами виникає можливість розвитку та широкомасштабних доповнень. На факультеті інформатики протягом останніх років постійно проводиться робота по впровадженню і вдосконаленню засобів електронного навчання в першу чергу з точки зору специфіки вивчення сучасного програмування.

В кваліфікаційній роботі розглядаються деякі можливості вдосконалення прийнятої на факультеті системи Moodle (останнім часом наведена вище абревіатура перетворилася на власну назву) . Не дивлячись на те що нові модулі додають нові можливості, вони не встигають за прогресом у навчальних методах і тому не завжди відповідають очікуванням Таким є модуль активності, який дозволяє студентам рецензувати роботи одне одного. Хоч цей модуль і має можливість для групової здачі роботи, він містить багато процесів які
повинні бути виконані вручну, як от пропонована оцінка або призначення рецензентів. Також у стандартній версії Moodle є усього 4 формати відображення курсу, що не покриває усі потреби.

Виходячи з описаного вище ***за мету даної роботи*** були поставлені розроблення модулів доповнення до системи Moodle.

Перше завдання полягало у вдосконаленні календарної системи ведення курсу. Був запропонований і реалізований новий формат відображення курсу – тижневий – із закріпленням за датою початкового тижня календарних термінів академічних активностей.

Практика вивчення програмування, застосована на факультеті передбачає взаємне рецензування студентами навчальних проектів [2], яке набуло особливої специфіки у випадку виконання групових начальних проектів. Тому друге завдання полягало в доповненні модуля рецензування студентських робіт новими можливостями:

1. Створення командного простору для роботи з наданням відповідних повноважень і прав членам команди і рецензентам;

2. автоматичного призначення рецензентів викладачем, як при роботі у команді, так і одноосібно

3. присвоєння команді капітана викладачем

4. прямого оцінювання роботи капітаном своїх підлеглих

5. прямого оцінювання роботи команди і окремого студента рецензентом

Об’єктом дослідження є система електронного навчання Moodle.

Мета роботи зумовила наступне ***наукове завдання:***

1. Ознайомитись із системою Moodle та можливостями для виконання групових студентських проектів у ній.

2. Виконати аналіз наявних проблем модуля рецензування з перспективи студента та викладача.

3. Розглянути можливості API Moodle та способи розробки нових модулів активності та форматів курсів.

4. Розробити нові модулі за допомогою програмних засобів.

5. Сформулювати висновки та надати рекомендації щодо впровадження створених модулів у навчання та їх подальшого розвитку.

Робота складається з трьох розділів.

У першому розділі розглядається система Moodle та можливості, які вона надає студентам та викладачам для навчання. Також оглядаються проблеми модуля рецензування та способи їх вирішення.

Другий розділ присвячено можливостям розробки нових модулів за допомогою програмних засобів. Розглянуто можливості API Moodle та основні типи модулів.

Третій розділ присвячено розробці та реалізації нових модулів мовами програмування PHP та JavaScript.

За результатами роботи було модернізовано користувацький інтерфейс для модуля рецензування, розширено його функціонал, а також створено два нових модулі для здачі робіт та один модуль формату курсу.

РОЗДІЛ 1: СИСТЕМА MOODLE ТА ЇЇ МОЖЛИВОСТІ. МОДУЛЬ РЕЦЕНЗУВАННЯ

1.1 Система електронного навчання Moodle

Moodle - це безкоштовна відкрита система управління навчанням (LMS - Learning Management System), призначена для створення та управління електронними курсами, змістом та оцінюванням студентів [3]. Moodle була розроблена як інструмент для навчання онлайн, який дозволяє педагогам створювати цифрові матеріали та дистанційні курси, що дозволяє студентам вчитися віддалено у відповідності зі своїми індивідуальними потребами.

Moodle надає широкі можливості для організації навчального процесу, такі як створення завдань, електронних ресурсів, дискусійних форумів, онлайн-тестів та багато іншого. Крім того, Moodle є дуже гнучкою системою, яка може бути налаштована під конкретні потреби та вимоги навчального закладу, викладачів та студентів. Вона також підтримує різноманітні мови та має активну спільноту розробників, яка постійно додає нові функції та покращує її функціональні можливості.

Moodle - це акронім, який розшифровується як "Modular Object-Oriented Dynamic Learning Environment". Цей термін відображає основні особливості системи Moodle, такі як модульність, об'єктно-орієнтованість та динамічність, які дозволяють надати гнучкі та інтерактивні можливості для навчання та сприяти активному залученню студентів.

У системі Moodle плагін (англ. plugin) - це додатковий модуль, який дозволяє розширити функціональні можливості основної системи [4]. Це може бути будь-який додатковий функціонал, наприклад, новий тип завдання, інструмент для роботи з мультимедіа, додаткові засоби для інтерактивного навчання, аналітичні засоби для відстеження прогресу студентів та інше.

Оскільки Moodle є відкритою системою, то кожен може створювати свої власні плагіни, щоб додати до системи нові можливості. Це дозволяє викладачам та адміністраторам підлаштувати навчальну платформу під конкретні потреби навчального закладу чи курсу, а також дозволяє створити інноваційні та цікаві методи навчання. Багато плагінів для Moodle розробляються та підтримуються спільнотою користувачів, що забезпечує широкі можливості для розвитку та покращення системи. Ознайомитись та завантажити їх можна за посиланням - https://moodle.org/plugins/.

1.2 Огляд можливостей студентів для роботи у команді

Важливість колаборативних завдань важко переоцінити, адже вони мають багато переваг над індивідуальними, оскільки вони сприяють розвитку соціальної та комунікативної компетенцій студентів, покращують розуміння та прийняття різних поглядів, зміцнюють взаємодію між студентами та допомагають розвивати співпрацю та командну роботу.

Одна з основних переваг колаборативних завдань полягає у тому, що вони можуть допомогти студентам здобути нові знання та розвинути свої навички, що було б важко зробити в індивідуальній роботі. Крім того, колаборативні завдання дозволяють студентам взаємодіяти між собою та зі своїми викладачами, обмінюючись ідеями та думками, відповідати на запитання один одного та вчитися один від одного.

Інша важлива перевага колаборативних завдань полягає у збільшенні мотивації студентів до навчання, оскільки вони можуть почувати себе більш залученими до процесу навчання, бачачи, що їхні ідеї та знання є важливими для успішного виконання завдань.

Дивлячись на результати порівняння ефективності групових завдань та індивідуальних на прикладі курсу «Засоби групової розробки програмних систем» (групова робота) та виробничої практики (індивідуальна робота) на

факультеті Інформатики НаУКМА, можна стверджувати, що при збільшенні складності завдання, якість виконання також зросла разом із розвитком індивідуальних навичок [5].

Moodle надає студентам різні можливості для групової роботи, зокрема:

1. Створення груп: Викладач може створити групи для студентів та дозволити їм працювати в групах на протязі курсу.
2. Форуми: Студенти можуть обговорювати питання відкрито в форумах та обмінюватись інформацією зі своїми колегами з групи та курсу.
3. Завдання: Викладач може створювати завдання, які передбачають співпрацю студентів для досягнення спільного результату.
4. Ресурси: Викладачі можуть надавати студентам різноманітні ресурси для допомоги у груповій роботі, такі як матеріали для вивчення.

Загалом, Moodle надає студентам широкий спектр можливостей для групової роботи, що дозволяє їм співпрацювати, обмінюватись ідеями та досягати спільних результатів. Найбільш важливим та цінним є третій пункт – адже він вимагає найбільшої зацікавленості студента.

Однак при використанні стандартних модулів активності групова складова вкрай бідна і передбачає лише спільну здачу роботи та одну її версію на всіх учасників групи. Більше колаборативних можливостей можна знайти у плагінах створених іншими користувачами або зробити його самому. Одним із таких плагінів є модуль взаємного рецензування.

1.3 Опис модуля рецензування та його проблеми

Модуль взаємного рецензування робіт студентами був створений на факультеті Інформатики НаУКМА [2]. Він передбачає такі можливості:

1. Викладач розподіляє роботи для рецензування

2. Рецензент одержує доступ до проекту іншого студента
3. Виконавши рецензію, рецензент розмістить її на порталі
4. Рецензія стане доступною викладачеві та авторові проекту

Цей модуль має чудову ідею - рецензування дозволяє виявити помилки в роботі або вказати на кращі рішення. Особливо корисно, коли роботу переглядає та оцінює не лише викладач, а й інший студент, який може мати свій власний підхід до вирішення завдання. Часто студенти можуть навіть бути більш критичними та прискіпливими у своїй оцінці, ніж викладач [1].

Однак він має недоліки як для викладача, так і для студентів.

Із перспективи студента:

1. При рецензуванні зазвичай ставиться оцінка в межах визначеній викладачем, але плагін цього не передбачає, тому оцінку рецензент ставить прямо в рецензії, що може бути не бажаним, якщо автор не повинен знати оцінку поставлену рецензентом.

2. При роботі у команді капітан повинен оцінити внесок своїх підлеглих, але такої можливості немає, тож доводиться робити це як в попередньому пункті.

Із перспективи викладача:

1. При призначенні рецензентів кожен студент повинен бути призначений вручну, методом перетаскування іконки до слоту рецензованого, що при великій кількості студентів стає марудним та збільшує шанси здійснити помилку.

2. Для правильного доступу до робіт інших студентів, викладач має здійснити призначення двічі з однаковими параметрами, пов’язавши два модуля активності між собою: один - робота, інший - рецензія на роботу. Шанс помилки збільшується в рази.

3. У випадку якщо рецензій на курсі багато, а рецензентів за авторами потрібно закріпити на весь курс, час на призначення зростає ще більше разом із шансом помилитись.

4. При виконанні у групі представлення студентів при призначенні рецензентів викладачем є однаковим. Розподіл по групах виконати ще важче ніж індивідуальний.

5. Задля уникнення академічної недоброчесності потрібно уникати пар рецензент – автор, що є взаємними.

6. При виконанні у групі потрібно бути уважним аби не призначити пару рецензент – автор студентів, що знаходяться в одній команді.

7. Відсутня можливість закріпити на курсі капітана групи.

8. При оцінюванні запропоновані бали рецензентами та капітанами ніяк не висвітлюються.

Наведені вище проблеми вказують на необхідність розширення функціональних можливостей плагіну з метою поліпшення зручності проходження курсу як для студентів, так і для викладачі

1.4 Постановка задачі про призначення рецензентів

Щоб вирішити проблему призначення рецензентів, найбільш правильним варіантом є автоматизація цього процесу. Для цього необхідно встановити відповідні критерії та підібрати авторів та рецензентів, що задовольнять ці умови:

1. при виконанні індивідуально:
    a. пари НЕ можуть бути взаємними.
2. при виконанні у команді:
    a. пари НЕ можуть бути взаємними
    b. пари НЕ можуть бути *близькими**
    c. команда повинна містити всіх рецензентів із різних команд
    d. не всі студенти можуть мати команду, тож ми можемо або ігнорувати їх, або вважати всіх хто не має команди як окрему команду

(*близькими* вважаються студенти із однієї групи)

Отже маємо:

- m – загальна кількість студентів
- n – кількість груп
- k – кількість студентів що не належать до жодної з груп

$$\text{де } k \leq m, n \leq m$$

Реалізація алгоритму пошуку рішення буде наведена у третьому розділі мовою програмування JavaScript.

1.5 Організація команд та оцінювання

При виконанні групових завдань команди зазвичай обирають собі капітана, проте ані в Moodle, ані в модулі рецензування таких можливостей немає. Так само як і у викладача відсутня інформація про лідера команди.

Для цього потрібно створити графічний інтерфейс на основі існуючого для призначення рецензентів, щоб використати готові компоненти, а не створювати нові.

Також, для рецензії є цілком логічним, що рецензент ставить оцінку у межах визначених викладачем, але зараз так опція відсутня. Так само як і можливість капітанам ставити оцінки своїм товаришам по команді.

Для вирішення проблеми нам потрібно створити два нові підплагіни – для оцінки рецензента та для оцінки капітаном. Також потрібно, щоб виставлені бали були доступні викладачу при швидкому оцінюванні.

1.6 Тижневий формат електронного курсу

Нерідко кожен навчальний заклад містить свій календарний графік, хтось має два семестри, хтось три триместри, хтось має канікули посеред навчання, а хтось ні. Moodle має кілька форматів курсів, серед яких основні [6]:

Topics - курс поділяється на теми, кожна з яких містить відповідний зміст.

Weeks - курс поділяється на тижневі блоки, кожен з яких містить відповідний зміст.

Social - курс містить один головний форум на головній сторінці.

Single activity format - курс складається з однієї великої активності, наприклад, форуму, календаря або вікі.

Одним із найбільш популярних є тижневий формат курсу, який автоматично визначає дати тижнів, якщо встановити початкову дату курсу.
Але він не має порядкового номеру тижня, що може викликати труднощі з орієнтуванням у навчальних закладах, де навчання центрується навколо кількості тижнів.

У третьому розділі буде наведена його реалізація мовою програмування PHP.

РОЗДІЛ 2: МЕТОДИ ТА ІНСТРУМЕНТИ РОЗРОБКИ ПЛАГІНІВ MOODLE

2.1. Локальна інсталяція

Перед тим як почати створення нового плагіну або оновлення існуючого спочатку треба встановити Moodle локально. Щоб встановити Moodle локально, спочатку потрібно встановити на свій комп'ютер веб-сервер, такий як Apache або Nginx та одну з реляційних баз даних на вибір(MySQL, MariaDb, PostgresQl). Також потрібно встановити PHP відповідної версії в залежності від цільової версії Moodle на свій комп'ютер. Потім завантажити потрібну версію Moodle із сайту та розмістити у бажаному каталозі системи [7].

Існують і більш прості способи встановлення, як наприклад повний інсталяційний пакет XAMPP [8] або віртуальна машина Bitnami LMS powered by Moodle™ LMS [9].

Під час розробки плагінів був обраний останній варіант. Використання віртуальної машини дозволяє швидко та легко встановити та налаштовувати Moodle LMS на локальному комп'ютері. Вона також містить усі необхідні залежності та налаштування, що робить їх ідеальним варіантом для тестування та розробки платформи Moodle LMS у віртуальному середовищі без необхідності встановлювати та налаштовувати окремі компоненти вручну. Перевагою такого підходу є також наявність програми керування встановленими компонентами: веб-сервером та базою даних. Також був встановлений веб-інтерфейс для управління базами даних phpMyAdmin [10].

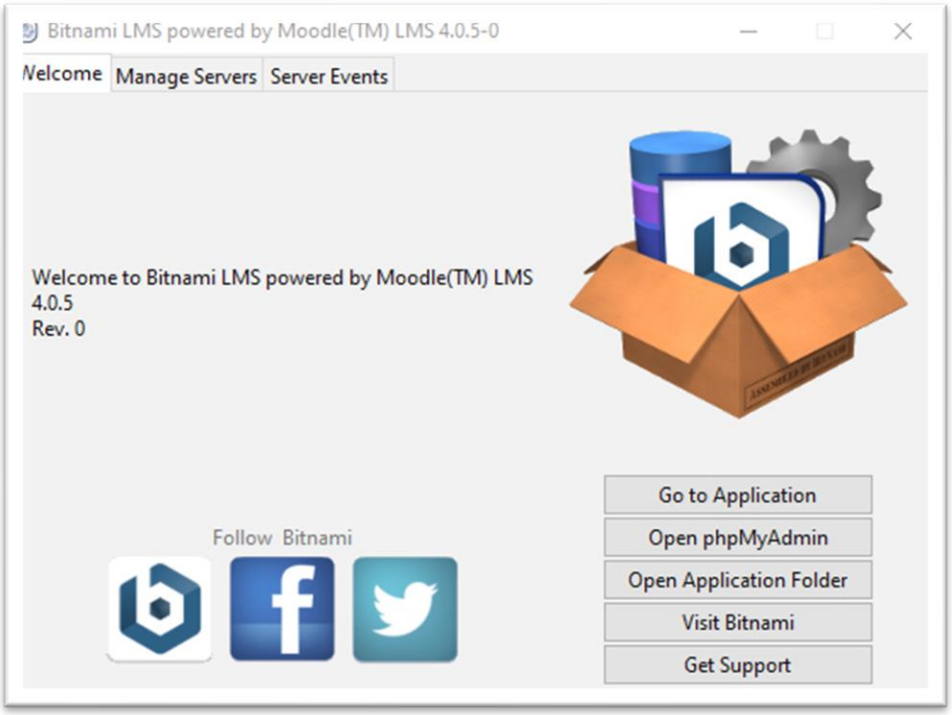

Рис. 2.1 Система керування встановленими компонентами від Bitnami

Після закінчення встановлення, для режиму розробки варто увімкнути режим відлагодження та вимкнути увесь кеш. Зробити це можна за шляхом *Site administration* > *Development* > *Debugging* [11]. Обираємо для відлагодження режим Developer, який виводить всі попередження та помилки всіх рівнів та дозволяє виявляти баги на ранній стадії.

2.2 Moodle API

Moodle API - це набір інтерфейсів програмування застосунків (англ. Application Programming Interface), які дозволяють нам інтегрувати різні модулі системи Moodle, а також розробляти плагіни, які розширюють функціональність Moodle [12].

Ця робота не буде описувати їх всі, оскільки їх дуже багато та не всі вони були використані при розробці нових плагінів. Натомість буде описано тільки ті API які використовувались найбільше.

2.2.1 Data Manipulation API

Data Manipulation API - це набір функцій, які дозволяють нам взаємодіяти з базою даних Moodle [13]. Цей API надає інструменти для створення, оновлення та видалення даних в Moodle. При розробці плагінів потрібно використовувати лише це API, адже тільки ці функції гарантують стабільну та безпечну роботу з будь-якою системою керування реляційними базами даних.

Для виконання запитів існує два способи – з безпосередніми написанням SQL(англ. Structured query language) [14] та використовуючи лише функції. При цьому писати SQL запит варто лише коли він складний, тобто містить JOIN або UNION чи IN. В усіх інших випадках використовувати функції буде ефективніше. Прикладом переваги використання готових функцій є запит з оператором IN – якщо у нас лише один параметр, то ефективніше буде використати WHERE. Для цього існує функція *get_in_or_equal($parameters)*. Функція повертає два значення, частину SQL запиту(IN або WHERE) та параметри. Далі отримані значення треба використати у функції *get_record(s)_sql($sql,$inparams)* для отримання записів із бази даних.

```
$getInOrEqual = $DB->get_in_or_equal($reviewsId);
$insql = $getInOrEqual[0];
$inparams = $getInOrEqual[1];
$sql = "SELECT * FROM {review_groups_captains} WHERE review $insql";
$group_captains = $DB->get_records_sql($sql,$inparams);
```

Головною перевагою використання Data Manipulation API є незалежність від обраної бази даних та захист від SQL ін’єкцій.

2.2.2 Data Definition API

Moodle Data Definition API - це набір функцій, який дозволяє нам керувати базою даних Moodle [15]. Цей API надає можливість створювати, змінювати та

видаляти таблиці бази даних Moodle, а також керувати індексами, приватними ключами та зовнішніми ключами.

На зміну способу коли схема бази визначалась за допомогою SQL для кожної бази даних окремо, прийшов XMLDB. XMLDB (XML DataBase) - це спосіб управління базами даних, який використовує XML файли для визначення схеми бази даних. XMLDB використовується для визначення структури таблиць бази даних та міграції даних між версіями плагінів Moodle [16].

Для кожної нової версії Moodle, XMLDB використовується для розширення або звуження бази даних, зміни структури таблиць, внесення змін до полів та індексів.

Для спрощення роботи із визначення та міграції схем бази даних рекомендується використовувати XMLDB Editor. XMLDB Editor - це графічний інтерфейс для редагування XML-файлів, які описують схему. Цей редактор дозволяє створювати, редагувати та експортувати схеми баз даних Moodle. За допомогою XMLDB Editor можна створювати нові таблиці, поля та індекси в базі даних, а також змінювати їх типи даних та інші параметри. Крім того, XMLDB Editor надає інтерфейс для міграції даних між різними версіями Moodle, що дозволяє зберігати дані при оновленні платформи Moodle.

Також при оновленні версії плагіну, нам потрібно скористатися опцією генерації PHP коду, який буде виконано якщо схема зазнала змін.

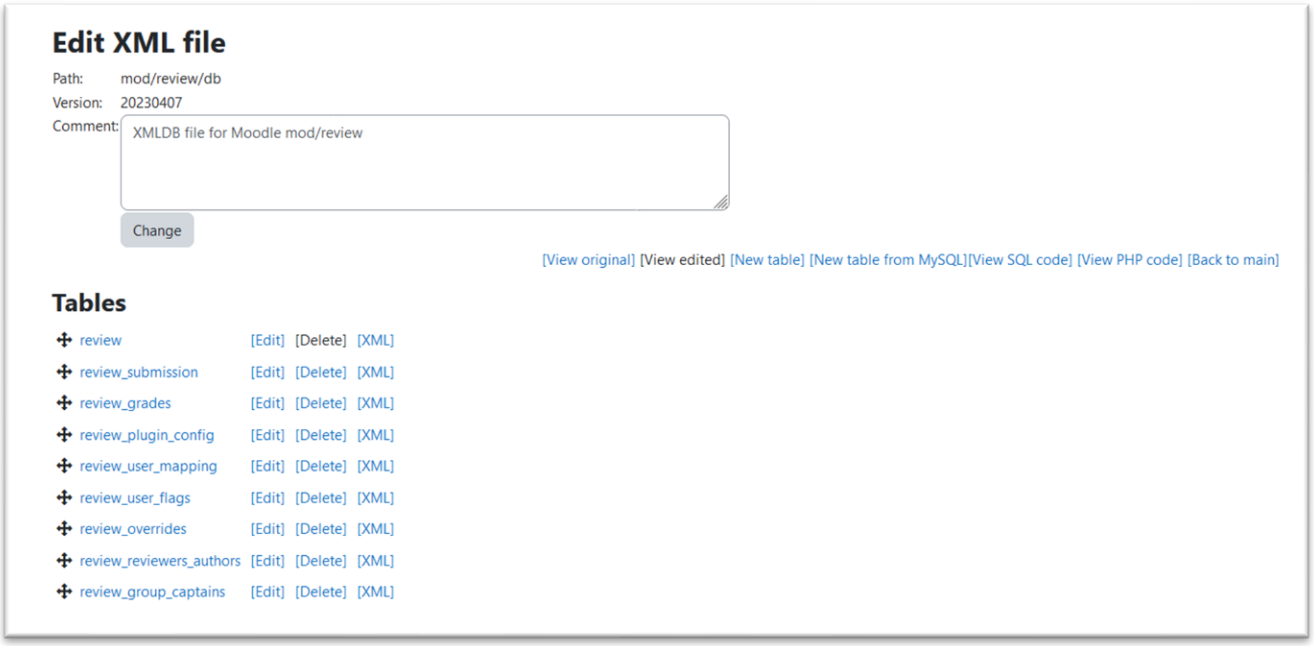

Рис. 2.2 Вигляд XMLDB Editor

2.2.3 Forms API

Forms API в Moodle - це набір функцій для створення HTML веб-форм за допомогою PHP [17]. Він дозволяє нам легко створювати форми для введення та редагування даних, вибирати значення зі списку, завантажувати файли, тощо.

Також Forms API дозволяє встановлювати обмеження на введення даних, використовувати валідатори для перевірки правильності введення даних, зберігати та відображати уже готові дані. Крім того, Forms API гарантує доступність(англ. Accessibility), що робить систему зручною для користування людям з обмеженими можливостями, а також спрощує розробку.

Основною функцією Form API є *addElement('type', 'name', 'label', 'restrictions')* яка безпосередньо додає бажаний елемент форми. Після створення елемента, йому можна додати правила валідації та обмеження. Обмеження можуть навіть залежати від стану інших елементів форми. Також можна обрати де буде відбуватись валідація, на стороні клієнта чи серверу. Основними функціями обмеження та валідації є:

- setDefault() – встановити значення поля за замовчуванням

- disabledIf() – вимкнути можливість вводу та редагування , якщо умова справджується
- hideIf() – повністю приховати поле, якщо умова справджується
- setType() – встановити тип даних, наприклад текстовий чи чисельний
- addRule() – додати правило валідації, наприклад required

У разі якщо дані не проходять валідацію, відповідне поле відобразить помилку додану у правилі з описом.

Також при використанні String API поле може містити підказку якщо, вона присутня у файлі локалізації та до елемента була застосована функція *addHelpButton()*.

2.2.4 String API

String API - це набір функцій для роботи з рядками. API допомагає нам забезпечити локалізацію та роботу з рядками, які можуть містити юнікод символи [18].

String API також надає функції для обрізки рядків, розділення рядків на складові, видалення пробілів, переведення рядків у нижній чи верхній регістр, форматування рядків і багато іншого.

Головна перевага використання String API – це локалізація, яка дозволяє надавати користувачам інтерфейс бажаною їм мовою. Для використання локалізації потрібно створити у директорії плагіну піддиректорію /lang/[language_code]/[plugin_name].php. Жоден плагін не може обійтись без директорії lang/en, оскільки вона мусить містити файл із рядком із назвою плагіну, для коректного встановлення та його роботи.

Щоб скористатися рядками визначеними у директорії /lang потрібно використати функцію *get_string(string_name, plugin_name)*, яка дозволяє отримати локалізований рядок залежно від обраної мови користувача.

2.3 Типи плагінів

2.3.1 Activity modules

Activity module - це тип плагіна в системі управління навчанням Moodle, який дозволяє нам створювати різноманітні завдання та інтерактивні вправи для студентів [19]. Викладачі можуть використовувати ці модулі для організації дистанційних занять, контролю знань, а також для стимулювання співпраці та взаємодії між студентами у групі шляхом залучення їх до групових проектів.

Модуль активності може містити різноманітні типи завдань, такі як тестування, форуми або завдання для відправки файлів чи тексту. Кожен модуль має свої параметри, які дозволяють викладачам налаштувати його під свої потреби. Також при першому встановленні плагіну у системі Moodle ці параметри встановлюються указаними розробником.

Модуль активності є одним з найбільш важливих та корисних типів плагінів у Moodle, оскільки вони дозволяють викладачам створювати інтерактивні та змістовні завдання для студентів, що сприяє покращенню якості навчання та розвитку навичок учасників навчального процесу.

Прикладом та одним з об’єктів дослідження цієї роботи є модуль активності review.

2.3.2 Assign submission plugins

Assign submission plugins - це плагіни, які дозволяють студентам здавати завдання для оцінювання в Moodle [20]. Плагіни можуть дозволяти завантажувати файли, вставляти текст або використовувати онлайн-інструменти для створення інтерактивного завдання. Також ці плагіни відповідають за відображення форм вводу та їх результат.

До стандартних плагінів здачі входять:

- File submissions - дозволяє завантажувати файли для завдань.
- Online text - дозволяє вводити у форму текст в різних форматах та додавати мультимедійні елементи до нього.

Плагін здачі робіт є важливою частиною іншого плагіну, модулю активності, адже без можливості здати роботу, модуль активності втрачає сенс.

2.3.3 Course format

Course format - це плагін, що відповідає за структуру та організацію навчального матеріалу, який відображається на курсі [21]. Він визначає розташування блоків, завдань, електронних ресурсів, мультимедіа та іншого матеріалу на сторінці курсу. Залежно від формату можуть бути доступні різні інструменти для роботи з матеріалом, наприклад, обговорення, тести, завдання тощо. Плагін формату курсу дозволяє викладачам створювати курси з різними структурами та зручними для викладання темами в залежності від потреб, а також легко мігрувати між ними за необхідності.

РОЗДІЛ 3: СТВОРЕННЯ ПЛАГІНІВ МОВАМИ ПРОГРАМУВАННЯ

3.1 Модуль рецензування

Оскільки не потрібно створювати весь модуль з нуля, а лише доробити функціонал, який можна реалізувати на стороні клієнта, редагувати потрібно буде лише п’ять фалів:

1. locallib.php – файл для написання глобальних функцій які можна використовувати у всьому плагіні.
2. assets/util.js – файл з клієнтським кодом.
3. assets/style.css – файл стилів CSS.
4. renderer.php – файл для графічного відображення елементів на сторінці.
5. version.php – файл для визначення версії плагіну.

Так як нових таблиць чи полів таблиць бази даних створювати не треба так само як і видаляти, то директорія /db залишається незмінною. Також оскільки ми не будемо зачіплювати роботу пов’язану із файлами при здачі робіт, то чіпати файл upgradelib.php також не потрібно. Варто зауважити, що деякі файли є обов’язковими, тому недбале їх редагування може призвести до небажаних наслідків. Ознайомитись із інформацією про типові файли можна за посиланням - https://moodledev.io/docs/apis/commonfiles.

3.1.1 Обробка даних

Перш ніж використати наші дані, нам спочатку потрібно їх отримати з бази даних, а щоб отримати їх з бази даних нам потрібно знайти метод який відповідає за рендер нашої веб-сторінки, яка надає інтерфейс для призначення рецензентів. При перегляді поточної сторінки бачимо, що у нашому URL є query parameter action=appointreviewers. В файлі locallib.php в методі *view($action = '', $args =*

array()) знаходимо потрібний нам блок із подальшою функцією-обробником дії призначення рецензента. Це функція *view_appoint_reviewers_page($args)*, яка повертає HTML сторінку.

```
} else if ($action == 'appointreviewers') {
    $o .= $this->view_appoint_reviewers_page($args);
} else if ($action == 'assigncaptains') {
    $o .= $this->view_assign_captains_page($args);
} else if ($action == 'saveappointreviewers') {
    $o .= $this->post_appoint_reviewers_page($args);
} else if ($action == 'saveassigncaptains') {
    $o .= $this->post_assign_captains_page($args);
```

Для отримання списку користувачів які записані на курс та мають право здавати роботу(тобто по суті мають роль студента) скористаємось Enrollment API [22]. Оскільки нам потрібно буде використати цей список у SQL запиті із JOIN обираємо *get_enrolled_sql()*.

```
list($esql, $params) = get_enrolled_sql(
  $this->context, "mod/review:submit");
```

Використовуємо отримане значення у наступному запиті, який поверне нам список студентів із полями ідентифікатора, імені, групи та призначеного рецензента, якщо такий є.

```
 $sql = 'SELECT CONCAT(u.id, ".", IFNULL(gm.groupid,"null")) rowkey, u.id,
u.firstname, u.lastname,  u.picture, u.imagealt, ra.authorid, gm.groupid
groupid,  gm.groupname groupname
          FROM (' . $esql . ') e
          JOIN {user} u ON e.id = u.id
     LEFT JOIN (SELECT _gm.userid, _gm.groupid, _g.name groupname
                FROM {groups_members} _gm
                JOIN {groups} _g ON _g.id = _gm.groupid
                WHERE _g.courseid = :courseid) gm
                ON e.id = gm.userid
     LEFT JOIN (SELECT *
                FROM {review_reviewers_authors}
                WHERE review = :reviewid) ra ON e.id = ra.reviewerid';
$params["courseid"] = $courseid;
$params["reviewid"] = $reviewid;
$users = $DB->get_records_sql($sql, $params);
```

Для того, щоб пов’язати роботу та рецензію нам потрібні всі інстанси активності review на курсі.

```
$reviews = $DB->get_records("review", array("course" =>
$courseid), '', "id, name, targetreviewid, enableaccess");
```

Після того як усі необхідні дані були отримані, перетворюємо їх у JSON за допомогою функції *json_encode()*.

```
$reviewsJson = json_encode($reviews);
$usersJson = json_encode($users);
```

Після цього можна переходити до клієнтської частини, де потрібно визначити групи та підгрупи студента, адже у Moodle перевірити або забезпечити, що одна група студентів є підгрупою іншої неможливо.

3.1.2 Створення та відображення груп

Гарним прикладом коли студент може бути присутнім одночасно в кількох групах є такий сценарій:

- На курсі декілька викладачів, кожен створив свою групу і працює з нею

- На курсі декілька потоків спеціальностей і викладач створив групи для кожної, щоб давати спеціалізовані завдання та відслідковувати їхній прогрес окремо
- Кожна з груп поділяється на академічні групи у відповідності до календарного графіку та самозапису студентів у групи
- Кожна з академічних груп ділиться на команди

Отже маємо при такому сценарії 3 або 4 шари груп.

Маємо справу з множинами, які містять в собі інші множини.

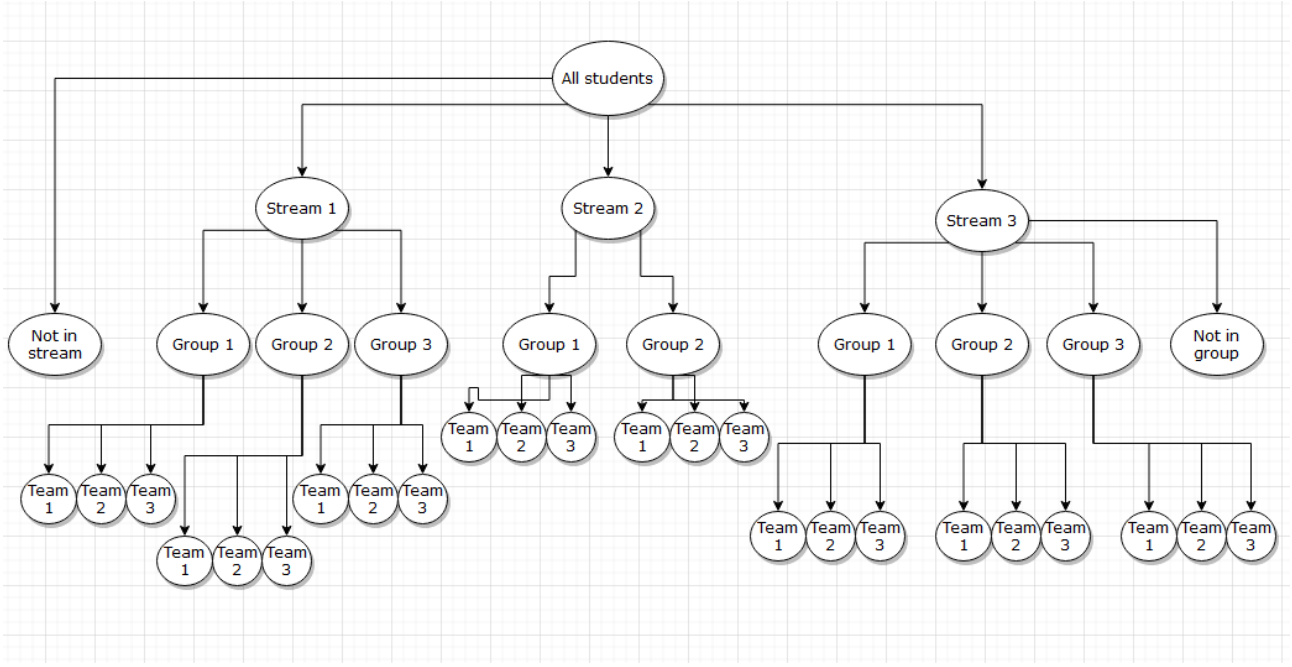

Рис. 3.1 Вигляд ієрархії груп

При перегляді діаграми одразу видно, що для вирішення задачі нам варто скористатись такою структурою даних як дерево. У якості кореня беремо множину всіх студентів. Також ми повинні врахувати, що не завжди всі студенти можуть бути у групах. Вершиною вузла позначатимемо ідентифікатор групи, а у значення записуватимемо студентів. Для студентів поза групою створюватимемо вузол із ідентифікатором батьківського вузла із знаком мінус.

Скористаємось готовою реалізацією дерева мовою програмування JavaScript із використанням функції-генератора для проходження по дереву [23] (Додаток А).

Для побудови дерева відсортуємо групи у порядку спадання. Потім пройдемо по ним циклом, визначаючи чи є поточна множина надмножиною груп менших за неї.

```
//ss is an object that contains pair: group id => array of subGroup ids
for (let i = 1; i < numberOfGroups; i++) {
    for (let k = i; k < numberOfGroups; k++) {
        if (isSuperset(new Set([...groups[i - 1].students]), new    Set([...groups[k].students]))) {
            const groupId = groups[i - 1].id;
            if (ss[groupId] !== undefined) {
                ss[groupId] = [...ss[groupId], groups[k].id]
            } else {
                ss[groupId] = [groups[k].id]
            }
        }
    }
}
```

Потім перевіримо чи наші підмножини повні і чи не потрібно створювати додаткові групи для студентів які є в надмножині, але відсутні в дереві нижче.

```
for (const [key, value] of Object.entries(ss)) {
    let inGroupStudents = [];
    value.forEach(groupId => {
        inGroupStudents = inGroupStudents.concat(groupMap[groupId].students);
    })
    const superSet = groupMap[key];
    const diff = difference(superSet.students.map(s => s.id), new
Set(inGroupStudents.map(s => s.id)))

    if (!!diff.length) {
        const students = diff.map(id => {
            const newStudent = {
                ...findStudentById(id),
                groupid: `-${key}`,
                groupname: `In "${superSet.name}" group, but none in nested`
            };
            ALL_STUDENTS.push(newStudent);
            UNIQUE_STUDENTS.set(id, newStudent);
            return newStudent
        });
        groupMap = {
            ...groupMap,
            [-key]: {
                id: `-${key}`,
                name: `In "${superSet.name}" group, but none in nested`,
                students: students
            }
        }
    }

}
```

Потім визначимо листки нашого дерева.

```
leaves.push(...difference(allGroupIds, new Set(Object.keys(ss))))
```

Визначаємо функцію-генератор, яка поверне двовимірний масив із групами на кожному рівні методом знаходження різниці між попереднім рівнем дерева та наступним. Функція йтиме знизу догори, отримуючи в якості попереднього рівня листки на початку, та список підмножин в якості параметра.

```
function* treeGenerator(subSets) {
    let prevNodes = leaves
    let nextNodes = []

    yield prevNodes
    while (!!Object.keys(subSets).length) {
        Object.keys(subSets).map(key => {
            if (difference(subSets[key], new Set(prevNodes)).length === 0) {
                nextNodes.push(key)
                delete subSets[key]
            }

        })
        yield nextNodes

        prevNodes = [...prevNodes, ...nextNodes]
        nextNodes = []
    }
}
```

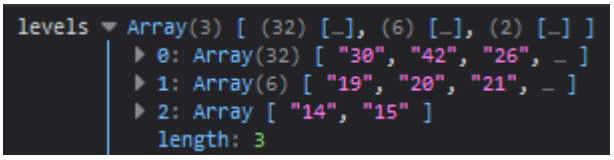

Рис. 3.2 Приклад результату функції

Потім ставимо у значення кореня дерева всіх студентів та робимо обхід нашого двовимірного масиву із рівнями дерева справа, вираховуючи перетин із глобальними надмножинами.

```
levels.reduceRight((prev, curr, index) => {
    prev.map(parent => {
        if (index === levels.length - 1) {
            curr.map(key => {
                GROUP_TREE.insert(parent, key, groupMap[key])
            })
        } else {
            intersection(levels[index], new Set(ss[parent]))
                .map(child => GROUP_TREE.insert(parent, child, groupMap[child]))
        }
    })
    return curr;
}, ["all"])
```

Після цього можна переходити до відображення дерева графічно. Графічний інтерфейс складається з допоміжних кнопок та трьох стовпців:

- Автор роботи
- Його рецензент
- Кандидат у рецензенти із числа авторів

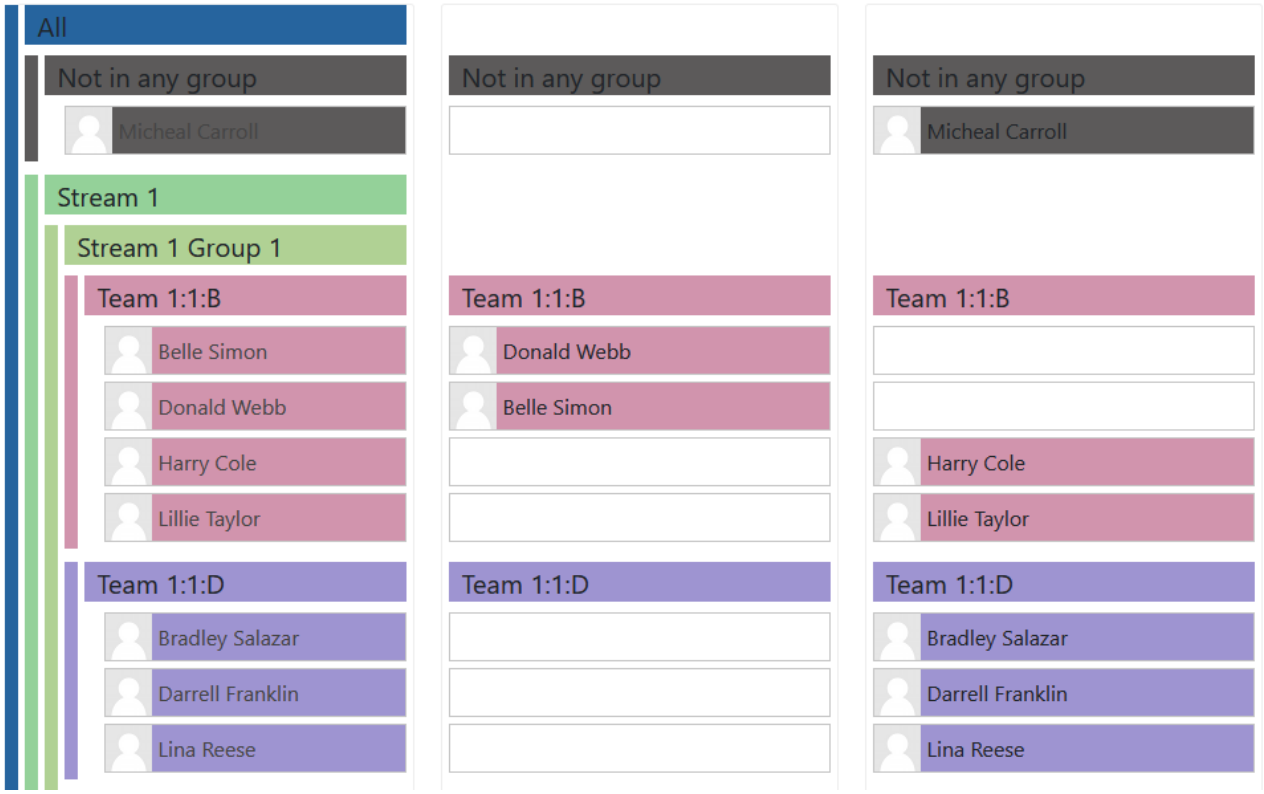

Рис 3.3 Вигляд стовпців із студентами

Завдяки тому, що піддерева є деревами ми можемо легко будувати ієрархію за допомогою рекурсивних функцій, а також будувати дерева поменше висмикуючи групи та роблячи область видимості для розподілу обмеженою.

```
function renderStudentBranch(node) {

    let html = "";

    if (node.isLeaf)
        html += renderStudents(node.value.students);
    else
        node.children.map(child => html += renderStudentBranch(child));

    const color = node.key === "null" ? "#5c5a5a" : selectColor(node.key)

    return `
                <div class="row branch">
                    <div class="col-auto vertical" style="background: ${color}"></div>
                    <div class="col">
                        <h5 class="group-heading" style="background:
${color}">${node.value.name}</h5>
                        ${html}
                    </div>
                </div>
        `;
}
```

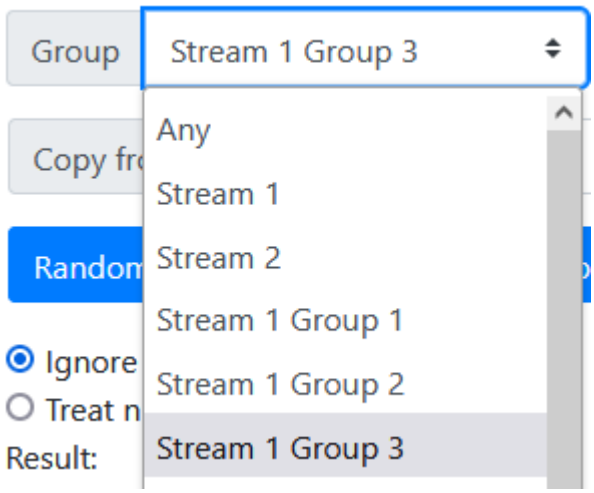

Рис. 3.4 Можливість обирати групи та звужувати розподіл

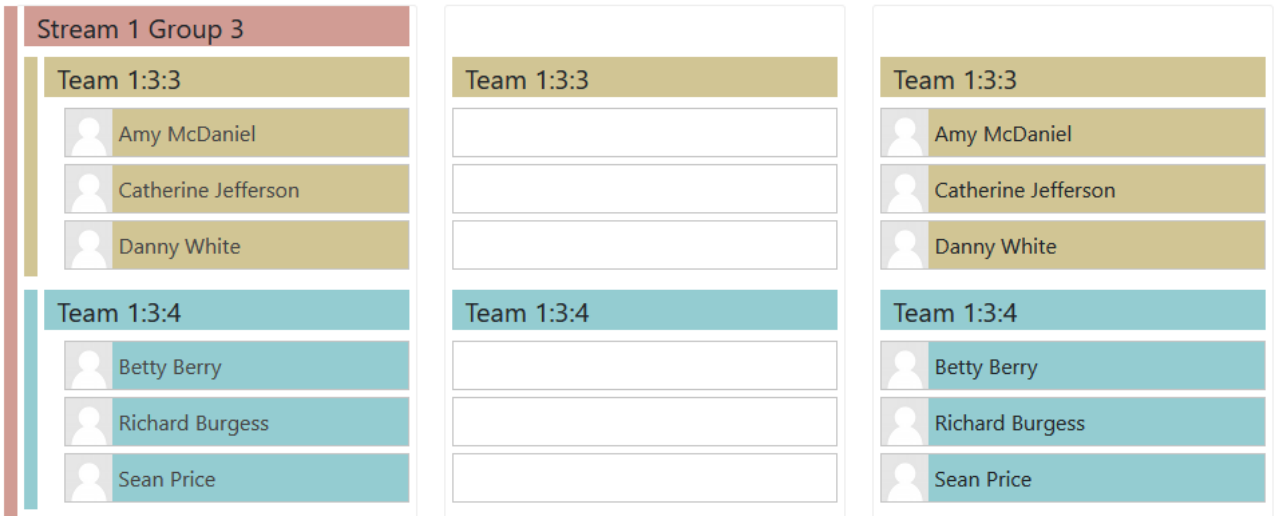

Рис. 3.5 Група з меншою областю розподілу

3.1.3 Випадковий розподіл рецензентів

Для випадкового розподілу нам необхідно перевірити чи маємо ми мінімальну кількість студентів(3). Потім створюємо кандидатів із авторів копіюючи масив, проходимо по масиву авторів, випадково обираючи кандидата та видаляючи його якщо він не взаємний або не є самим автором. Отримані пари рендеримо аналогічно до стовпця авторів, показаного раніше.

```
function randomAssign() {
    RESET_BUTTON.trigger("click")
    const currentRootNode = getCurrentRootNode();
    const studentsIds = currentRootNode.value.students.map(s => s.id);
    if (studentsIds.length < 3)
        return;
    const authorPool = [...studentsIds];
    const pairs = new Map();
    studentsIds.map((item) => {
        let author = authorPool.random()
        while ((author === item || pairs.get(author) === item) &&
authorPool.length !== 1)
            author = authorPool.random()
        pairs.set(item, author);
        authorPool.splice(authorPool.findIndex(i => i === author), 1)
    })
    pairs.forEach((key, value) => {
        assignAuthor(value, key)
    })
    renderColumns(currentRootNode)

}
```

Також необхідно створити кнопку та прив’язати до неї обробник події. Оскільки на сторінці вже використовується Bootstrap та jQuery, створимо кнопку використовуючи можливості цих бібліотек.

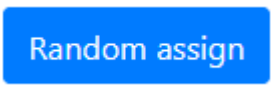

Рис. 3.6 Вигляд кнопки

Рис. 3.7 Вигляд стовпців після натискання кнопки

3.1.4 Груповий розподіл рецензентів

Поставлена нами задача є найбільш схожою на проблему стабільних шлюбів (англ. Stable Matching Problem або SMP) розв'язану у 1962 році Девідом Гейлом та Ллойдом Шеплі [24]. Враховуючи що чоловіки та жінки у нас є однією множиною ця задача могла б бути розв'язана як проблема стабільних сусідів [25], якби не додаткові умови, що кімнати у нас не є гарантовано двомісними та навіть не завжди однакові за розміром та уподобання співмешканців постійно змінюються в залежності від вибору інших сусідів.

Якщо допустити взаємні пари та можливість утворення пари із самим собою, то цю задачу можна розв'язати як проблему стабільних шлюбів із динамічними вподобаннями з використанням механізму пошуку та видалення

блокуючих пар [26]. Після знаходження розв'язку ми зможемо виконати умови пізніше.

Для вирішення задачі скористаємось алгоритмом Гейла-Шеплі для симетричних вподобань між жінками і чоловіками (Додаток Б). В якості умов нестабільності відповідностей беремо наявність в одній групі більш ніж одного студента з однієї групи та/або студента з тієї ж самої групи.

```
let data = galeShapley(flatLeafStudents, false);

while(true) {
    let [removedPairs, occ] = removePairs(data, leaves, flatLeafStudents);
    const length = removedPairs.length;
    const diversity = {}
    removedPairs.forEach(x => {
        data.delete(x)
        diversity[flatLeafStudents.find(o => o.id === x).groupid] = true
    })

    if (length === 0) {
        break
    } else if (length === 1) {
        const lastStudent = flatLeafStudents
            .find(o => o.id === removedPairs[0]);
        const restrictions =
            new Set(Object.keys(
                occ.find(x => lastStudent.groupid === x.group).map
            )
            .filter(o => o !== lastStudent.groupid))
        for (const group of occ) {
            if (group.group !== lastStudent.groupid) {
                const targetGroup = Object.keys(group.map);
                const diff = difference(targetGroup, restrictions);
                for (const proposition of diff) {
                    if (!diff.includes(lastStudent.groupid)) {
                        let tempTargetStud = group.map[proposition][0]
                        let tempData = data
                        tempData.delete(tempTargetStud)
                        tempData.set(tempTargetStud, lastStudent.id)
                        tempData.set(lastStudent.id, data.get(tempTargetStud))
                        const [removedPairs, _] =
                            removePairs(tempData, leaves, flatLeafStudents);
                        if (removedPairs.length === 0) {
                            data = tempData
                            break;
                        }
                    }
                }
                if (data.size === flatLeafStudents.length)
                    break
            }
        }
        if (data.size === flatLeafStudents.length) {
            break
        } else {
            data = galeShapley(flatLeafStudents, false);
        }
    } else if (Object.keys(diversity).length <= 2) {
        data = galeShapley(flatLeafStudents, false);
```

```
    } else {
        let newStudents = flatLeafStudents
            .filter(x => removedPairs
                .includes(x.id))
            .map(x => x.id)

        newStudents =
            updatePreferencesDynamically(flatLeafStudents, data, leaves)
            .filter(x => newStudents.includes(x.id))
            .map(x => ({
                ...x,
                partnerId: null,
                preferences: x.preferences.filter(o => newStudents.includes(o))
            }))

        let newData = galeShapley(newStudents, true);
        newData.forEach((key, value) => {
            data.set(key, value)
        })
    }
}

data = shuffleReviewers(data, leaves)
data.forEach((key, value) => {
    assignAuthor(value, key)
})
```

Після отримання розв’язку виконуємо тасування рецензентів всередині розподілених груп, тим сам нівелюючи взаємні пари.

Для такого типу задач нерідкісні і метаевристичні підходи до вирішення. Спробуємо знайти рішення використавши метод локального пошуку.

```
const candidatesByGroup = {}
leaves.map((leaf, index) => {
    candidatesByGroup[leaf.id] = new Set(candidates[index])
})
const randoms = randomPairs(candidatesByGroup, flatLeafStudents);
const occ = checkOcc(randoms, leaves, flatLeafStudents);
const fil = occ.map(i => i.arr.length === new Set(i.arr).size)
    .filter(val => val === false);
if (fil.length === 0 && randoms.size === flatLeafStudents.length) {
    randoms.forEach((key, value) => {
        assignAuthor(value, key)
    })
    renderColumns(node)
    RESULT_TEXT.text("Solution found!")
    return;
}
```

```
function randomPairs(candidatesInp, studentsInp) {
    const candidates = {...candidatesInp}
    const students = [...studentsInp]
    const pairs = new Map();
    const assigned = new Set([])
    let hasSolutions = true
    while (students.length > 0 && hasSolutions) {
        const randomStudent = students.random();
        if (!candidates[randomStudent.groupid].size)
            break
        while (candidates[randomStudent.groupid].size > 0) {
            const randomAuthor = Array
                .from(candidates[randomStudent.groupid])
                .random();
            if (assigned.has(randomAuthor)) {
                candidates[randomStudent.groupid].delete(randomAuthor)
                if (candidates[randomStudent.groupid].size === 0) {
                    hasSolutions = false
                    break;
                }
            } else {
                assigned.add(randomAuthor)
                candidates[randomStudent.groupid].delete(randomAuthor)
                students.splice(students
                    .findIndex(i => i.id === randomStudent.id), 1)
                pairs.set(randomStudent.id, randomAuthor)
                break;
            }
        }

    }

    return pairs
}
```

При порівнянні результатів швидкості роботи алгоритмів для різних наборів даних ми обираємо алгоритм локального пошуку, оскільки для більшого набору даних результати кращі й при їх збільшенні результати тільки покращуватимуться, а для менших результати є незначущими(1мс та 3 мс).

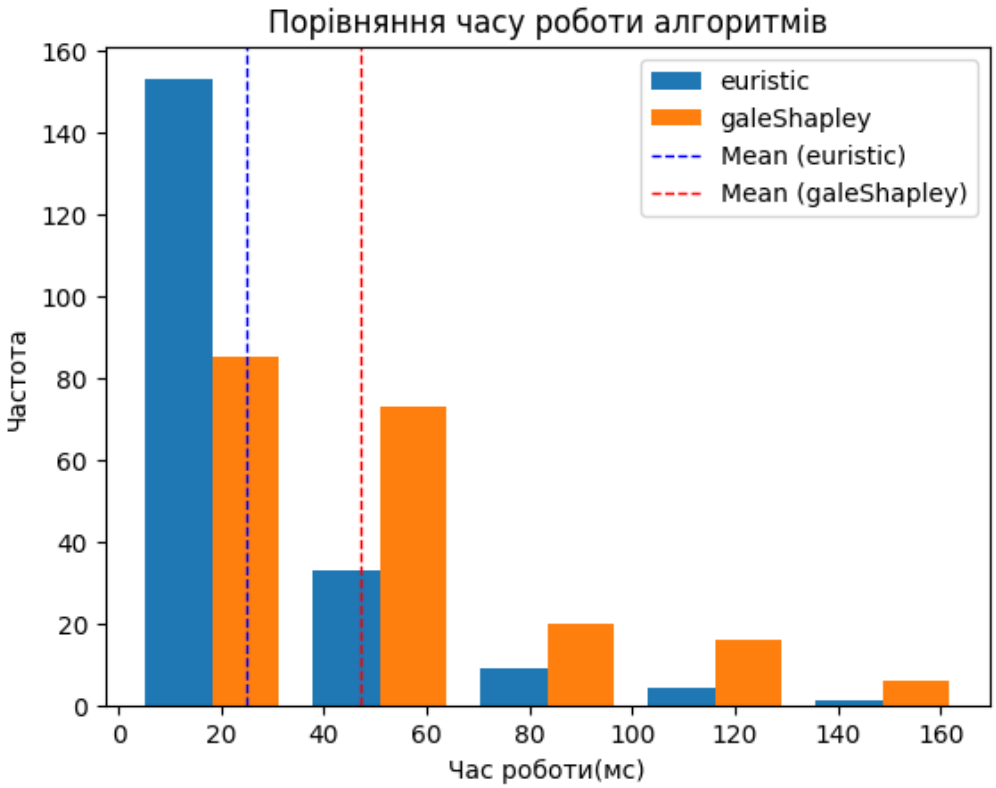

Рис. 3.8 Порівняння часу роботи алгоритмів, к-сть випробувань 200 (к-сть студентів 100, к-сть груп 32, мінімум у групі 1, максимум 5)

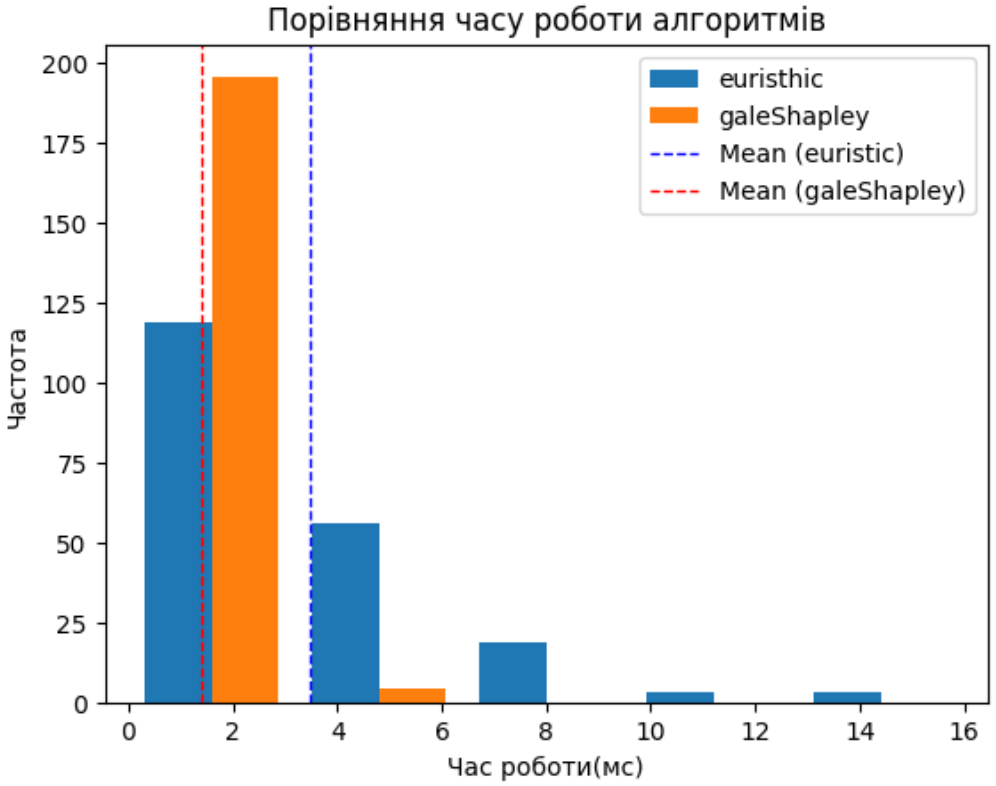

Рис. 3.9 Порівняння часу роботи алгоритмів, к-сть випробувань 200 (к-сть студентів 15, к-сть груп 5, мінімум у групі 2, максимум 4)

3.1.5 Додаткові елементи та можливості

Було додано можливість очистити всіх рецензентів та скопіювати їх з інших інстансів завдання, що дозволяє не повторювати набір рецензентів вручну для підтримання одних і тих же рецензентів на весь курс. Також було створено текстове поле яке інформує про успіх стан операцій(очищення, розподіл, чи є оптимальні рішення).

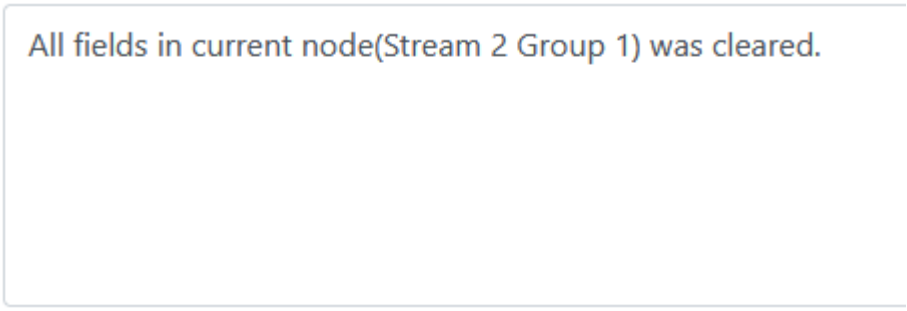

Рис. 3.10 Поле інформування

3.2. Плагіни здачі робіт

3.2.1 Студентське оцінювання

Оскільки нам потрібно створити плагін з нуля, скористаємось порадою з офіційного сайту Moodle і візьмемо за референс стандартний плагін onlinetext. Копіюємо його та перейменовуємо всі відповідні файли.

У файлі version.php встановлюємо версію у форматі *дата+номер.* У файлі settings.php встановлюємо параметр *maxgrade = 100,* який за замовчуванням буде пропонуватись встановити викладачу при налаштуванні завдання вперше, як максимальна оцінка студента.

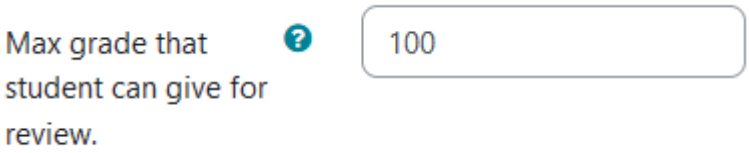

Рис. 3.11 Параметр при налаштуванні здачі викладачем

Далі нам потрібно створити файл з описом таблиць бази даних. Оскільки це перша версія файлу, нам не потрібно думати про сумісність з попередніми файлами, тому створюємо файл db/install.xml.

```
<TABLES>
    <TABLE NAME="reviewsubmission_studgrade" COMMENT="Info about studgrade
submission">
        <FIELDS>
            <FIELD NAME="id" TYPE="int" LENGTH="10" NOTNULL="true"
SEQUENCE="true"/>
            <FIELD NAME="review" TYPE="int" LENGTH="10" NOTNULL="true"
DEFAULT="0" SEQUENCE="false"/>
            <FIELD NAME="submission" TYPE="int" LENGTH="10" NOTNULL="true"
DEFAULT="0" SEQUENCE="false"/>
            <FIELD NAME="studgrade" TYPE="float" NOTNULL="false"
SEQUENCE="false" COMMENT="The grade for this student grade submission."/>
        </FIELDS>
        <KEYS>
            <KEY NAME="primary" TYPE="primary" FIELDS="id" COMMENT="The unique
id for this studgrade submission."/>
            <KEY NAME="review" TYPE="foreign" FIELDS="review" REFTABLE="review"
REFFIELDS="id" COMMENT="The review instance this student grade submission
relates to."/>
            <KEY NAME="submission" TYPE="foreign" FIELDS="submission"
REFTABLE="review_submission" REFFIELDS="id" COMMENT="The submission this student
grade submission relates to."/>
        </KEYS>
    </TABLE>
</TABLES>
```

Для створення поля в якому студент повинен ввести бал, у файлі locallib.php в методі *get_form_elements()* додаємо форму за допомогою Forms API.

```
if ($submission) {
    $studgradesubmission = $this->get_studgrade_submission($submission->id);
    if ($studgradesubmission) {
        $data->studgrade = $studgradesubmission->studgrade;
    }
}


$mform->addElement('text', 'studgrade', get_string('gradeproposition',
'reviewsubmission_studgrade'));
$mform->addRule('studgrade', 'Must be numeric value.', 'numeric');
$mform->addRule('studgrade', 'This field is required', 'required');
```

Так само для викладача потрібно створити форму в якій він може увімкнути опцію такої здачі та змінити максимальний бал. В тому ж файлі знаходимо метод *get_settings()* та додаємо поля.

```
if ($this->review->has_instance()) {
    $defaultmaxgrade = $this->get_config('maxgrade');
} else {
    $defaultmaxgrade = get_config('reviewsubmission_studgrade', 'maxgrade');
}

$name = get_string('maxgrade', 'reviewsubmission_studgrade');
$mform->addElement('text', 'reviewsubmission_studgrade_maxgrade', $name,
'min=0');
$mform->addHelpButton('reviewsubmission_studgrade_maxgrade',
    'maxgrade',
    'reviewsubmission_studgrade');
$mform->hideIf('reviewsubmission_studgrade_maxgrade',
'reviewsubmission_studgrade_enabled', 'notchecked');
$mform->addRule('reviewsubmission_studgrade_maxgrade', 'Must be numeric value.',
'numeric');
$mform->setDefault('reviewsubmission_studgrade_maxgrade', $defaultmaxgrade);
```

Тепер потрібно обробити оцінку запропоновану студентом та зберегти її у базі даних. Перевіряємо чи оцінка більше нуля та не перевищує максимальну встановлену викладачем.

```
if ($this->review->has_instance()) {
    $defaultmaxgrade = $this->get_config('maxgrade');
} else {
    $defaultmaxgrade = get_config('reviewsubmission_studgrade', 'maxgrade');
}
if($grade < 0)
    $grade = 0;
if($grade > $defaultmaxgrade)
    $grade = $defaultmaxgrade;

if ($studgradesubmission) {
    $studgradesubmission->studgrade = $grade;
    return $DB->update_record('reviewsubmission_studgrade',
$studgradesubmission);
} else {
    $studgradesubmission = new stdClass();
    $studgradesubmission->studgrade = $grade;
    $studgradesubmission->review = $this->review->get_instance()->id;
    $studgradesubmission->submission = $submission->id;
    return $DB->insert_record('reviewsubmission_studgrade',
$studgradesubmission) > 0;
}
```

3.2.2 Капітанське оцінювання

Для плагіну з капітанами аналогічно встановлюємо параметр за замовчуванням та поля для налаштуванням завдання викладачем.

Цього разу схема бази даних вимагатиме також створення асоціативної таблиці із капітанами.

```
<TABLES>
    <TABLE NAME="reviewsubmission_capgrade" COMMENT="Info about capgrade
submission">
        <FIELDS>
            <FIELD NAME="id" TYPE="int" LENGTH="10" NOTNULL="true"
SEQUENCE="true"/>
            <FIELD NAME="review" TYPE="int" LENGTH="10" NOTNULL="true"
DEFAULT="0" SEQUENCE="false"/>
            <FIELD NAME="submission" TYPE="int" LENGTH="10" NOTNULL="true"
DEFAULT="0" SEQUENCE="false"/>
            <FIELD NAME="capid" TYPE="int" LENGTH="10" NOTNULL="true"
DEFAULT="0" SEQUENCE="false"/>
            <FIELD NAME="groupmemberid" TYPE="int" LENGTH="10" NOTNULL="true"
DEFAULT="0" SEQUENCE="false"/>
            <FIELD NAME="capgrade" TYPE="float" NOTNULL="false" SEQUENCE="false"
COMMENT="The grade for this student grade submission from captain of group."/>
        </FIELDS>
        <KEYS>
            <KEY NAME="primary" TYPE="primary" FIELDS="id" COMMENT="The unique
id for this capgrade submission."/>
            <KEY NAME="capid" TYPE="foreign" FIELDS="capid"
REFTABLE="review_groups_captains" REFFIELDS="userid" COMMENT="The captain of
group"/>
            <KEY NAME="groupmemberid" TYPE="foreign" FIELDS="groupmemberid"
REFTABLE="groups_members" REFFIELDS="userid" COMMENT="Non-captain member of
group"/>
            <KEY NAME="review" TYPE="foreign" FIELDS="review" REFTABLE="review"
REFFIELDS="id" COMMENT="The review instance this student grade submission
relates to."/>
            <KEY NAME="submission" TYPE="foreign" FIELDS="submission"
REFTABLE="review_submission" REFFIELDS="id" COMMENT="The submission this student
grade submission relates to."/>
        </KEYS>
    </TABLE>
    <TABLE NAME="review_groups_captains" COMMENT="Map captains to groups for
review">
        <FIELDS>
            <FIELD NAME="id" TYPE="int" LENGTH="10" NOTNULL="true"
SEQUENCE="true"/>
            <FIELD NAME="review" TYPE="int" LENGTH="10" NOTNULL="true"
SEQUENCE="false"/>
            <FIELD NAME="groupid" TYPE="int" LENGTH="10" NOTNULL="true"
DEFAULT="0" SEQUENCE="false"/>
            <FIELD NAME="userid" TYPE="int" LENGTH="10" NOTNULL="true"
DEFAULT="0" SEQUENCE="false"/>
        </FIELDS>
        <KEYS>
            <KEY NAME="primary" TYPE="primary" FIELDS="id"/>
            <KEY NAME="review" TYPE="foreign" FIELDS="review" REFTABLE="review"
REFFIELDS="id" COMMENT="The review instance"/>
            <KEY NAME="groupid" TYPE="foreign" FIELDS="groupid"
REFTABLE="groups" REFFIELDS="id" COMMENT="The group"/>
            <KEY NAME="userid" TYPE="foreign" FIELDS="userid"
REFTABLE="groups_members" REFFIELDS="userid" COMMENT="The captain of group"/>
        </KEYS>
    </TABLE>
</TABLES>
```

Для присвоєння командам капітанів перевикористаємо графічний інтерфейс для призначення рецензентів. Також обмежимо кількість слотів до одного, аби уникнути неоднозначності.

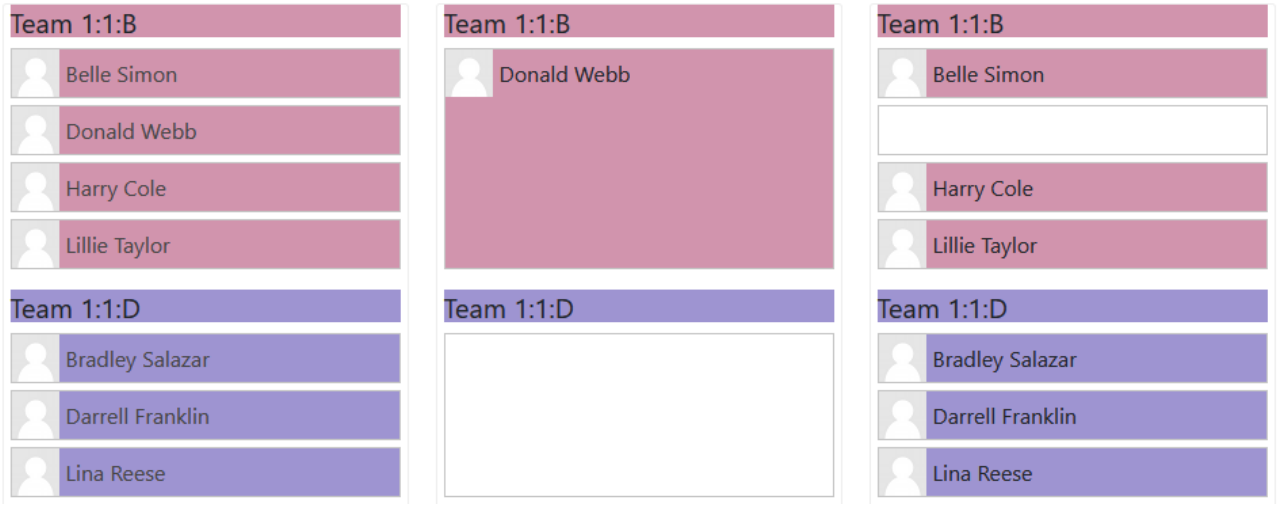

Рис. 3.12 Інтерфейс капітанів

Також як і в студентському оцінюванні необхідно додати поля для капітана при здачі роботи в яких він може виставити свої бали для підлеглих.
Важливо переконатись, що поля для капітана бачить лише капітан.

```
$userid = $USER->id;
$isCaptain = $DB->get_record('review_groups_captains', array('review' => $this-
>review->get_instance()->id, 'userid' => $userid));

if ($isCaptain) {
    $submissionid = $submission ? $submission->id : 0;
    $groupmembers = $DB->get_records('groups_members', array('groupid' =>
$isCaptain->groupid), '', 'userid');
    $groupmembers = array_filter($groupmembers, function ($x) use ($userid) {
        return $x->userid != $userid;
    });
    if (!isset($data->capgrade)) {
        $data->capgrade = '';
    }
    print_object($groupmembers);
    list($insql, $params) = $DB->get_in_or_equal(array_column($groupmembers,
'userid'));
    $sql = "SELECT * FROM {user} WHERE id $insql";
    $groupmembers = $DB->get_records_sql($sql, $params);
    foreach ($groupmembers as $groupmember) {
        $mform->addElement('text', 'capgrade-' . $groupmember->id,
            get_string('gradeproposition', 'reviewsubmission_capgrade',
$groupmember->firstname . ' ' . $groupmember->lastname));
        $mform->addRule('capgrade-' . $groupmember->id, 'Must be numeric
value.', 'numeric');
        $mform->addRule('capgrade-' . $groupmember->id, 'This field is
required', 'required');
    }

    if ($submission) {
        $capgradesubmission = $this->get_capgrade_submission($submission->id);
        if ($capgradesubmission) {
            foreach ($capgradesubmission as $record) {
                $data->{'capgrade-' . $record->groupmemberid} = $record-
>capgrade;
            }
        }
    }


}
```

Рис. 3.13 Обов'язкові поля видні лише для капітана команди при здачі роботи

Останній важливий пункт плагінів здачі, який є спільний для обох форматів – це відображення оцінок викладачу на панелі оцінювання.

```
$sql = "SELECT rsg.studgrade, reviewer.firstname, reviewer.lastname FROM
{review} r
        JOIN {review_reviewers_authors} rra ON r.id = rra.review AND
rra.authorid = :authorid
        JOIN {review_submission} rs ON rs.review = r.id AND rs.userid =
rra.reviewerid
        JOIN {reviewsubmission_studgrade} rsg ON rsg.submission = rs.id
        JOIN {user} reviewer on reviewer.id = rra.reviewerid
        WHERE r.targetreviewid = :reviewid";

$params["reviewid"] = $submission->review;
$params["authorid"] = $submission->userid;
$studgrade = $DB->get_record_sql($sql, $params);
if ($studgrade) {
    $o .= "<h6><i>Reviewer $studgrade->lastname $studgrade->firstname graded
this work for $studgrade->studgrade points</i></h6>";
} else {

    $targetreview = $DB->get_record(
        "review",
        ["targetreviewid" => $submission->review]
    );

    if ($targetreview) {
        $sql = "SELECT reviewer.firstname, reviewer.lastname FROM {review} r
        JOIN {review_reviewers_authors} rra ON r.id = rra.review AND
rra.authorid = :authorid
        JOIN {user} reviewer on reviewer.id = rra.reviewerid
        WHERE r.targetreviewid = :reviewid";

        $params["reviewid"] = $submission->review;
        $params["authorid"] = $submission->userid;
        $reviewer = $DB->get_record_sql($sql, $params);
        if($reviewer){
            $o .= "<h6><i>Reviewer $reviewer->lastname $reviewer->firstname has
not graded this assignment yet</i></h6>";
        } else {
            $o .= "<h6><i>This student does not have reviewer yet</i></h6>";
        }
    }
}
```

Рис. 3.14 Вигляд підказок для викладача з балами студентів та капітанів

3.3 Плагін формату курсу

За аналогією з плагіном здачі, створюємо новий курс формату на основі існуючого – тижневого формату. Щоб закріпити початковий тиждень нам потрібно створити поле у меню редагування курсу. Для цього у файлі lib.php знаходимо метод *course_format_options()* та додаємо до списків опцій та опцій редагування такі списки:

```
'startweek' => array(
    'default' => 1,
    'type' => PARAM_INT,
),

'startweek' => array(
    'label' => new lang_string('startweek','format_regularweeks'),
    'element_type' => 'text',
    'help' => 'startweek',
    'help_component' => 'format_regularweeks',
),
```

Рис. 3.15 Вигляд нового поля з опцією стартового тижня

У тому ж файлі знаходимо метод *get_default_section_name()* та змінюємо відображення секцій використовуючи новий параметр формату – *startweek.*

```
$startweek = $this->get_course()->startweek;

// We subtract 24 hours for display purposes.
$dates->end = ($dates->end - 86400);

$dateformat = get_string('strftimedateshort');
$weekday = userdate($dates->start, $dateformat);
$endweekday = userdate($dates->end, $dateformat);
return 'Week '. ($startweek + $sectionnum - 1).' ('.$weekday.' -
'.$endweekday.')';
```

Рис. 3.16 Новий вигляд секцій із номерами тижня

3.4 Аналіз і оцінка тривалості проектів. PERT

Оскільки активність рецензії є багатоетапною та вимагає одночасної участі викладача та студентів потрібно проаналізувати тривалість виконання одного завдання аби вкластись у календарний план.

Для цього використаємо техніку оцінки тривалості проектів PERT для побудови розкладу етапів та аналізу критичного шляху [27].

Формула PERT: (О + (4 * Н) + П) / 6

Дати завершення вузлів активності для модуля рецензування:

- Оптимістична – 7 днів
- Песимістична - 9 днів

- Найймовірніша - 8 днів

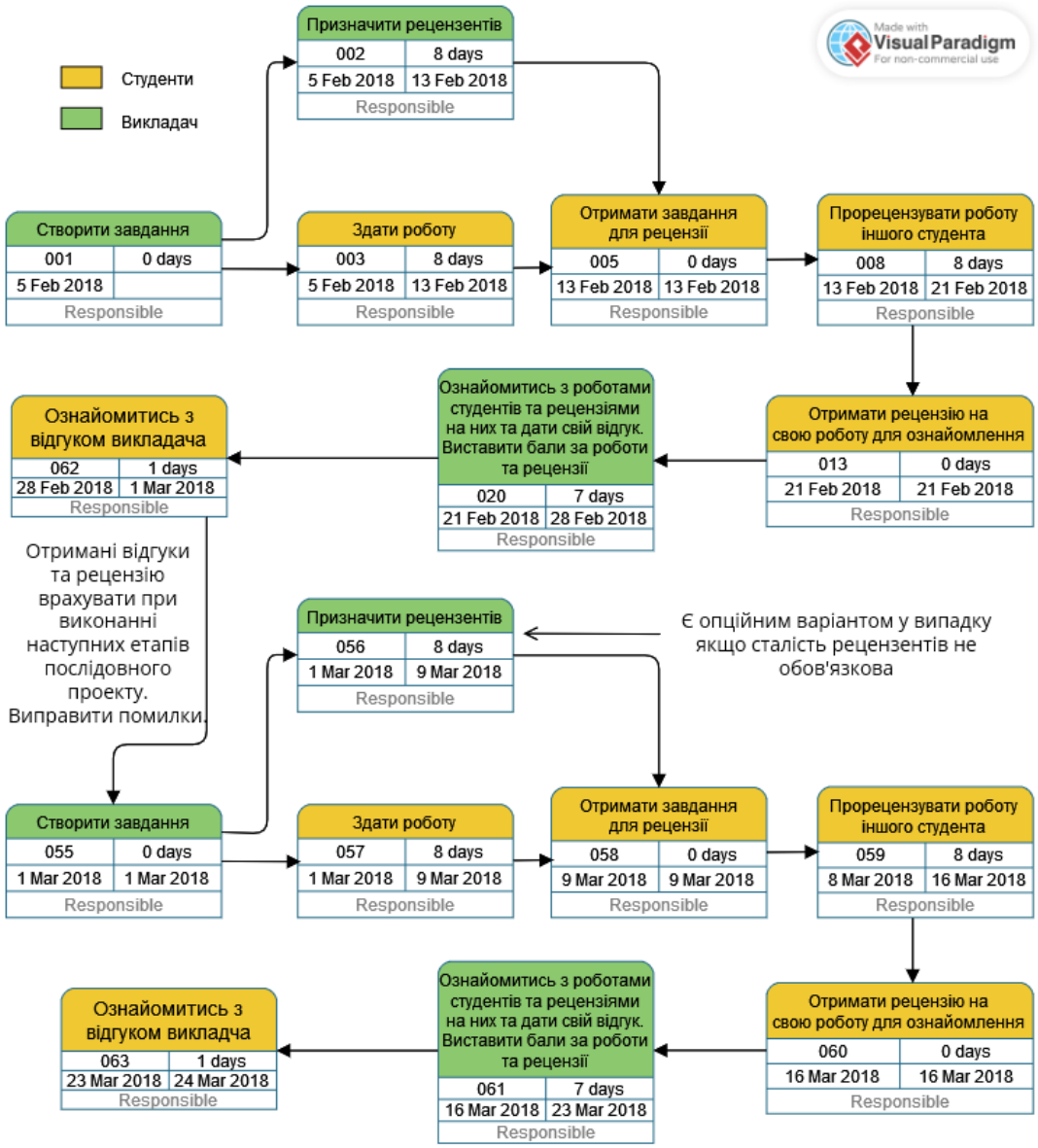

Рис. 3.17 PERT діаграма для двох етапів проекту із застосуванням активності рецензування

Оскільки вузли «Призначити рецензентів» та «Здати роботу» мають однакову тривалість(як для песимістичного варіанту так і для оптимістичного), наш критичний шлях у діаграмі є будь-яким і складає 27 днів для одного етапу проекту.

ВИСНОВКИ

У роботі було оглянуто можливості колаборативного навчання за допомогою системи електронного навчання Moodle та її недоліки. Завдяки отриманим знанням було вдосконалено структуру календарного плану і запропоновано командну організацію груп студентів.

Методами аналізу і оцінки тривалості проектів (PERT) запропоновано плани проходження активностей курсу з урахуванням зв'язків між окремими етапами виконання робіт і їх взаємного рецензування.

Створено та інтегровано до системи модуль підтримки групового рецензування, який забезпечив розподіл рецензентів за заданими критеріями з дотриманням вимог академічної доброчесності.

Створено модуль взаємного оцінювання членів команди і оцінювання членів команди їх капітанами.

Подальші вдосконалення передбачаються в напрямку збору статистики проходження курсу за командами, групами та спеціальностями; створення електронного простору курсу, включаючи робочі зошити окремих студентів з подальшим вдосконалення системи управління правами доступу.

ВИКОРИСТАНІ ДЖЕРЕЛА

1. Бублик В. В. Шляхами дистанційної освіти та електронного навчання / Бублик В. В. // Наукові записки НаУКМА. Комп'ютерні науки. - 2018. - Т. 1. - С. 4-9. — URL: https://ekmair.ukma.edu.ua/handle/123456789/14639
2. Бублик В. В. Розвиток колаборативних навчальних середовищ / Бублик В. В., Дроздович Н. Ю. // Наукові записки НаУКМА. - 2012. - Т. 138: Комп'ютерні науки. - С. 76-79. – URL: https://ekmair.ukma.edu.ua/handle/123456789/1919
3. Moodle [Електронний ресурс] - https://docs.moodle.org/402/en/About_Moodle
4. Moodle Plugin [Електронний ресурс] - https://moodle.com/faq/what-is-a-moodle-plugin/
5. Бублик В. В. Особливості впровадження навчальної групової розробки програмних систем / Бублик В.В., Афонін А.О., Борозенний С.О. // Наукові записки НаУКМА. - 2008. - Т. 86 : Комп'ютерні науки. - С. 73-77. – URL: https://ekmair.ukma.edu.ua/handle/123456789/6041
6. Moodle Course Formats [Електронний ресурс] - https://docs.moodle.org/402/en/Course_formats
7. Installing Moodle [Електронний ресурс] - https://docs.moodle.org/402/en/Installing_Moodle
8. XAMPP [Електронний ресурс] - https://docs.moodle.org/402/en/Complete_install_packages_for_Windows
9. Bitnami Moodle [Електронний ресурс] - https://bitnami.com/stack/moodle/virtual-machine
10. phpMyAdmin [Електронний ресурс] - https://www.phpmyadmin.net
11. Moodle Debug Mode [Електронний ресурс] - https://docs.moodle.org/402/en/Debugging
12. Moodle API Guides [Електронний ресурс] - https://moodledev.io/docs/apis
13. Data manipulation API [Електронний ресурс] - https://moodledev.io/docs/apis/core/dml
14. SQL [Електронний ресурс] - https://www.w3schools.com/sql/

15. Data definition API [Електронний ресурс] - https://moodledev.io/docs/apis/core/dml/ddl
16. XMLDB [Електронний ресурс] - https://docs.moodle.org/dev/XMLDB_Defining_one_XML_structure
17. Forms API [Електронний ресурс] - https://moodledev.io/docs/apis/subsystems/form
18. String API [Електронний ресурс] - https://docs.moodle.org/dev/String_API
19. Activity Modules [Електронний ресурс] - https://moodledev.io/docs/apis/plugintypes/mod
20. Assign submission plugins [Електронний ресурс] - https://moodledev.io/docs/apis/plugintypes/assign/submission
21. Course format plugin [Електронний ресурс] - https://moodledev.io/docs/apis/plugintypes/format
22. Enrollment API [Електронний ресурс] - https://moodledev.io/docs/apis/subsystems/enrol
23. JavaScript Tree [Електронний ресурс] - https://www.30secondsofcode.org/articles/s/js-data-structures-tree/
24. Gale D., Shapley L. S. College Admissions and the Stability of Marriage. *The American Mathematical Monthly*. 1962. Т. 69, № 1. С. 9. URL: https://www.eecs.harvard.edu/cs286r/courses/fall09/papers/galeshapley.pdf
25. Boehmer N., Elkind E. Stable Roommate Problem with Diversity Preferences. *Twenty-Ninth International Joint Conference on Artificial Intelligence and Seventeenth Pacific Rim International Conference on Artificial Intelligence {IJCAI-PRICAI-20}*, м. Yokohama, Japan, 11–17 лип. 2020 р. California, 2020. URL: https://arxiv.org/pdf/2004.14640.pdf
26. Alimudin A., Ishida Y. Matching-Updating Mechanism: A Solution for the Stable Marriage Problem with Dynamic Preferences. *Entropy*. 2022. Т. 24, № 2. С. 263. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8871443/
27. Kelley J. E. Critical-Path Planning and Scheduling *Operations Research*. 1959. Т. 9, № 3. С. 160 - 173. URL: https://mosaicprojects.com.au/PDF/PM-History_Critical_Path_Planning_&_Scheduling_Kelley_and_Walker_1959.pdf

ДОДАТОК А: КЛАСИ TreeNode ТА Tree

```
class TreeNode {
    constructor(key, value = key, parent = null) {
        this.key = key;
        this.value = value;
        this.parent = parent;
        this.children = [];
    }

    get isLeaf() {
        return this.children.length === 0;
    }

    get hasChildren() {
        return !this.isLeaf;
    }
}

class Tree {
    constructor(key, value = key) {
        this.root = new TreeNode(key, value);
    }

    * preOrderTraversal(node = this.root) {
        yield node;
        if (node.children.length) {
            for (let child of node.children) {
                yield* this.preOrderTraversal(child);
            }
        }
    }

    * postOrderTraversal(node = this.root) {
        if (node.children.length) {
            for (let child of node.children) {
                yield* this.postOrderTraversal(child);
            }
        }
        yield node;
    }

    insert(parentNodeKey, key, value = key) {
        for (let node of this.preOrderTraversal()) {
            if (node.key === parentNodeKey) {
                node.children.push(new TreeNode(key, value, node));
                return true;
            }
        }
        return false;
    }

    remove(key) {
        for (let node of this.preOrderTraversal()) {
            const filtered = node.children.filter(c => c.key !== key);
            if (filtered.length !== node.children.length) {
                node.children = filtered;
                return true;
            }
        }
        return false;
    }

    find(key) {
        for (let node of this.preOrderTraversal()) {
            if (node.key === key) return node;
        }
        return undefined;
    }
}
```

ДОДАТОК Б: АЛГОРИТМ ГЕЙЛА-ШЕПЛІ

```
function galeShapley(students, startPrefs = false) {
    let men = startPrefs ? students.slice() : updatePreferences(students)
    let women = men.slice()
    const priorities = new Map();
    const freeMen = men.slice();
    let engagements = [];
    for (let student of students) {
        priorities.set(student.id, 0);
    }

    while (freeMen.length > 0) {
        let man = freeMen.shift();
        let womanId = man.preferences[priorities.get(man.id)];
        let woman = women.find(woman => woman.id === womanId);
        let currentPartner = engagements.find(pair => pair.woman.id == woman.id)

        if (currentPartner === undefined) {
            engagements.push({man, woman});
            man.partnerId = woman.id;

        } else {
            let currentMan = currentPartner.man;
            let womanPreference = woman.preferences.indexOf(man.id);
            let currentManPreference = woman.preferences.indexOf(currentMan.id);

            if (womanPreference < currentManPreference) {
                engagements = engagements
                    .filter(pair => pair.man.id !== currentMan.id);
                engagements.push({man, woman});
                currentMan.partnerId = null;
                man.partnerId = woman.id;
                freeMen.push(currentMan);
            } else {
                freeMen.push(man);
            }
        }
        priorities.set(man.id, priorities.get(man.id) + 1);
    }
    const res = new Map()

    engagements.forEach(x => {
        res.set(x.man.id, x.woman.id)
    })

    return res;
}
```