**ЗМІСТ**

ВСТУП.....2
1. АНАЛІЗ ІСНУЮЧІХ ТЕСТОВИХ СИСТЕМ, ЇХ ОСОБЛИВОСТІ ТА ПОРІВНЯННЯ.....4
1.1 Тестування, дослідження його переваг та недоліків.....4
1.2 Види тестів и принципи їх складання.....6
1.3 Аналіз та огляд існуючіх тестових систем знанн.....13
1.4 Дослідження найбільш розповсюдженої тестової системи.....15
2 РОЗРОБКА СИСТЕМИ ТЕСТУВАННЯ.....24
2.1 Постановка задачі.....24
2.2 Серверні засоби для реализации системи тестування.....25
2.3 Установка Apache и настройка файрвол.....25
2.4 Налаштування файрволу для контролю веб-трафіку.....25
2.5 Налаштування MySQL.....26
2.6 Налаштування PHP.....27
2.7 Тестування роботи PHP.....28
2.8 Структура тестового порталу.....29
2.9 Адміністраторський блок.....37
2.10 Операторський блок.....47
2.11 Студентський блок.....53
2.12 Модуль контролю студентів.....63
3 АПРОБАЦІЯ РОЗРОБЛЕНОЇ СИСТЕМИ.....64
3.1 Огляд адмін-модулю.....64
3.2 Оператор-модуль.....69
3.3 Студент-модуль.....70
ВИСНОВОК.....74

# ВСТУП

**Актуальність теми**. На сучасному етапі розвитку освіти в Росії все більше уваги приділяється впровадженню ефективних форм контролю знань. Однією з таких форм є проведення тестування. Це найбільш стандартизований і об'єктивний метод контролю і оцінювання знань, умінь і навичок випробуваного, який позбавлений таких традиційних недоліків інших методів контролю знань, як неоднорідність вимог, суб'єктивність екзаменаторів, невизначеність системи оцінок і т. П. Актуальність теми дисертаційної роботи визначається необхідністю розробки системи тестування, що усуває недоліки розглянутих систем, що забезпечує реалізацію адаптивних методів тестування, що спирається при розробці тестів на теоретичні основи як класичної, так і сучасної теорії тестування.

**Метою** роботи є створення новітньої системи контролю знань, що дозволяє своєчасно отримувати інформацію про якість навчального процесу, необхідну для прийняття управлінських рішень. Для досягнення мети необхідно вирішити такі завдання:

1. Проаналізувати існуючих систем тестування, виявити їх переваги та недоліки;
2. Розробити систему тестування студентів; Визначити її структури і функціональне призначення її модулів. Вибрати найбільш відповідні та сучасні технології реалізації.
3. Протестувати розроблену систему в конкретній предметній області.

**Об’єктом дослідження** є розробка системи тестування з застосуванням сучасних серверних рішень.

**Предметом дослідження** є система тестування з застосуванням сучасних серверних рішень.

**Методи дослідження** які були використані при виконанні роботи: дедуктивний, індуктивний, метод аналізу та синтезу.

**Джерела дослідження:**

- Аванесов ВС «Наукові проблеми тестового контролю знань», видавництво «Вища школа», 2014
- Василенко Т.Д., Нікішина В.Б. «Психологія. Тестовий контроль знань », видавництво " Владос ", 2005
- Прокоф'єва Н.О. «Методи контролю знань при комп'ютерному навчанні», видавництво «Освіта і віртуальність», 2005

**Наукова новизна** одержаних результатів полягає в :

- Вперше запропоновано структуру web системи тестування студентів з вбудованою моделлю дляконтролю знань, яка відрізняється від існуючих наявністю технології відеоконтролю кожного студента, та обгрунтована її реалізація.
- Розроблено нові алгоритми перевірки об’єктивності проведеного тестування.

**Практичне значення** одержаних результатів:

Розроблені в дисертації системи і алгоритми проведення тестування дозволяють перевести його на якісно вищий рівень, коли з'являється можливість індивідуалізувати підхід до кожного учня на підставі виявленої структури знань, організувати контроль і підвищити точність вимірювань рівня знань за рахунок зменшення впливу списування на результат.

**Апробація результатів** магістерської роботи:

- IX науково-технічна конференція студентів та молодих вчених навчально-наукового інституту інформаційних технологій Державного Університету Телекомунікацій, 5 грудня 2019 року.

**Публікації**:

- Збірник тез “Сучасні інфокомунікаційні технології” – “Розробка системи для тестування студентів”, ст. 207

# 1. АНАЛІЗ ІСНУЮЧІХ ТЕСТОВИХ СИСТЕМ, ЇХ ОСОБЛИВОСТІ ТА ПОРІВНЯННЯ

## 1.1 Тестування, дослідження його переваг та недоліків

З давніх часів тестування було одним з найбільш зручних способів перевірки і оцінки знань. Тестування - з англ. test – випробування або перевірка. Саме слово «тест» має французьке коріння и означає посудину, ака використовується в медицині для проведення різніх дослідів. У наш час це слово вживають, коли треба пройти перевірку, пробу, випробування. Багато людей звикло, что тестові завдання представлені в виді питань з декількома варіантами відповідей. Однако предками сучасних тестів були випробування, які треба було пройти претенденту на посаду в якості перевірки. Так, ще около двох тисяч років тому в Китаї кандидатів на високі посади відбирали з допомогою тестів, а в стародавнбому Єгипті посвячення в жерці могло відбутися лише після серії випробувань.

На сьогоднішній день тестування проникло в багато сфер діяльності людини. В освіту, роботу, медицину і т.д. Тести дуже популярні в Америці, їх використовують для відбору найбільш гідніх кандидатів . Бізнесмени вважають цю систему дуже зручною, тому що вона досить легко вірішити проблему компетенції кадрів. Фахівці вважають, що тести дуже надійні в прогнозах несприятливих результатів (наприклад, тести на те чи варто вам займатіся певної видом діяльності). Розвіненість будь-якіх якостів або здібностей визначається не так точно. Тому за допомогою тестового відбору легше обмежити коло претендентів.

Тести можуть класифікуватися за різними ознаками. Тести поділяються на тести досягнень і особістісні. До першого типу відносяться тести інтелекту або тести шкільної успішності, тести на творчість, тести здатності до певної діяльності, сенсорні та моторні тести. До інших - тести установки, на захоплення, на темперамент або характерологічні тести. Однак не всі тести (наприклад, тести розвитку, графічні тести) можна класифікувати за цими ознаками. Залежно від того, наскільки в

результаті тестування суб'єктивність дослідника впливає на тест, розрізняють тести об'єктивні и суб'єктивні.

До об'єктивних тестів відносяться більшість тестів досягнень та психо-фізіологічні тести, до суб'єктивних - проектні тести. Цей поділ деякою мірою схожий з поділом на прямі и непрямі тести, які різняться в залежності від того, знають чи не знають люди які проходять тестування значення та мета тесту.

Для проектних тестів типова ситуація, коли випробуваний не знає про справжню мету дослідження. При виконанні завдання проективних тестів немає "правильних" відповідей. В залежності від наявності в тесті мовного компонента ,тести діляться на вербальні та невербальні. Вербальним, наприклад, є тест на запас слів, а невербальний – це тест, завдання в якому подано у вигляді малюнків, рисунків, символів тощо. За структурою розрізняються тести можуть бути прості, тобто елементарні, результатом якіх может бути одна відповідь, та тести складні, які можуть складатися з декількох підтестів, по кожному з яких буде поставлена оцінка. При цьому можуть бути враховані і загальні оцінки. Кожен вид тестування має свои плюси та мінуси.

Плюси тестування:

- Можлівість Проводити масові вимірювання знань без значних затрат.

- Можлівість встановлювати рівень знань випробовуваного по предмету в цілому и по його окремих розділах.

- Всі випробовувані знаходяться в однакових умовах.

- Об'єктивність оцінювання знань, в більшості видів тестів, перевіряючий не може впливати на результат тесту .

Недоліки тестування:

- У різних груп випробовуваних різний рівень знань, тому тести повинні враховувати це.

- Можлівість випробуваного вгадати відповідь.

- Результати, одержані тим хто його складав, в результаті тестування, хоч і мають в собі інформацію про прогалини в знаннях по конкретним питанням, але не дозволяють судити про причини цих прогалин.

- Розробка якісного тесту - довгий, складний та дорогий процес. Стандартні набори тестів для більшості дисциплін постійно змінюються, а уже розроблені зазвічай є застарівшими.

- Тест не дозволяє перевіряти та оцінюваті якісні характеристики випробовуваного пов'язані з творчістю.

- Широта охоплення різноманітних тем в тестуванні має і зворотний бік. Випробовуваний , на відміну від усного а різноманітних або письмовий тестування, не має достатньо часу для глибокого аналізу питання.

**1.2 Види тестів и принципи їх складання**

Відбір и структура тесту залежить від того, що цікавить перевіряючого даної групи осіб. Завдання тесту за своєю суттю представляють для випробуваного проблему. Відповідь на питання  це усунення сумнівів, вагань, невизначеності в ситуації, що робиться з метою отримання нових, більш точних знань.

Кожне завдання тесту містіть опис певної "ситуації", взятої з природи, життя, і т.д. Воно може бути представлено на різніх "мовах", а саме: вербальною, мовою символів, графіків, рисунків і тому подібне. Опис ситуації завжди неповний і тому, перед тестом ставиться "вимога" описати цю ситуацію більш точно, вікористовуючі з цією метою "інформацію для вирішення", яка є в опису ситуації, тексті завдання, а також залучаючи "зовнішню до завдання інформацію "- тобто відомі випробуваному наукові факти, закономірності, експерименти  и т.п.

Буває багато видів тестів. Можна виділити:

- Тести досягнення;
- Крітеріальною-орієнтовані тести, які дозволяють перевірити рівень знань, умінь і навичок в повному обсязі.

- Нормативно-орієнтовані тести, вони порівнюють випробовування один з одним за рівнями та учбовим досягненням;
- Атестаційні тести, які дозволяють підвести підсумок степені навченості;
- Тести прогнозування

Тести використовують для повної оцінки стану випробовуваних, наприклад, до початку процессу їх розвитку, здібностей до конкретного предмету, оцінки уровня навченості, уровня досягнені в даній області знань.

Слід відразу зазначити, що багато видів перерахованих тестів поки не знайшли свого застосування в вузах, їх розробка і застосування попереду. Більше того, можливо створення багатофункціональних тестів, коли один і той же тест при використанні іншої методики його застосування, методики роботи з результатами тестування, може бути використовуватись з різними цілями, наприклад, для контролю, діагностики, навчання і прогнозування його результатів. Майже кожен тест дозволяє робити висновки про ефективність технології, на основі якої проводилося навчання.

Останнім часом все більше визнання отримує новий вид тестів – критеріальною -орієнтовані ( "criterion-referenced test"). Як і будь-який педагогічний тест, критеріальною-орієнтований тест являє собою систему завдань, що дозволяє виміряти рівень навчальних знань. Та головна їх особливість полягає в тому, що критеріальною-орієнтовані тести зіставляють рівень індивідуальних навчальних досягнень з повним обсягом знань, умінь і навичок, які мають бути засвоєні випробовуваними.

Зазвичай критеріальною-орієнтовані тести використовуються викладачами для розподілу студентів (учнів) на декілька груп за рівнем їх знань. У найпростішому випадку це дві групи - які засвоїли і не засвоїли необхідний навчальний матеріал. У тестах освітній стандарт показаний у вигляді критеріального бала - мінімального тестового бала, який має отримати студент, щоб вважатися людиною,яка засвоїла

матеріал.

Критеріальною-орієнтовані тести вирішують проблему співвідношення набраних тестових балів з традиційними педагогічними оцінками.

Критеріальною-орієнтовані тести допомагають вирішити наступні проблеми:

- Зіставлення з необхідним стандартом освіти оцінок окремих студентів або навчальних груп;
- Оцінка відсотку навчального матеріалу, засвоєної студентами;
- Вибір певної технології навчання на основі отриманих результатів;
- Відбір випробовуваних, які досягли необхідного рівня кваліфікації.

На відміну від критеріальною-орієнтованих, тести досягнень є зручним засобом для швидкого і ефективного виявлення рівня професійної підготовки студента і рівнів навченості матеріалу учнів. Тести професійних досягнень досить специфічні, призначаються для конкретних професій, тому область їх застосування вважається обмеженою. Можна виділити дві групи тестів професійних досягнень:

Перша група вимагає від випробуваного виконання деяких контрольних завдань, що моделюють задачу в професійній діяльності, на практично-семінарських заняттях;Друга група це набір завдань, які виконуються студентами на практиці.

Випробовувані, в залежності від умов поставлених завдань, можуть відповідати на них як письмово так і усно. Викладач в окремих випадках може проводити опитування в формі розмови або інтерв’ю. Всі ці способи характеризуються однією ознакою: з їх допомогою отримують ту інформацію, яка закладена в словесних повідомленнях опитуваних. При бесіді та інтерв'ю завдання обговорюються в певній послідовності, яка передбачена конструкцією педагогічного тесту.

- Методику розробки тестів досягнень можна розбити на декілька структурних етапів: Підбір типових ситуацій, які виникають у професійній діяльності вчителя, з одного боку, вони повинні бути синтезовані і об'єднані навколо певної практичної проблеми, що має багатосторонній і цілісний характер, а з іншого,вони повинні бути переведені

на мову практичних дій, практичних ситуацій;

- Розробка методів перекладу ситуацій на мову педагогічних завдань;
- Розробка різних варіантів вирішення педагогічних завдань;
- Визначення цінності (вага) різних рішень по кожній конкретній педагогічної задачі;
- встановлення правильної послідовності пред'явлення системи завдань, що становлять
- Тест досягнень;
- Експериментальна перевірка складеного тесту. Аналіз професіографічних даних і додаткових відомостей про навчальні програми, методах, формах, засобах навчання показав, що для розробки тестів досягнень цього недостатньо, потрібно ще й наявність спеціальної системи професійних знань і умінь.

У первинному варіанті тест досягнень має надмірну кількість завдань, частина яких після першої експериментальної перевірки забирається. Про необхідність мати надлишок завдань в попередньому варіанті говорять всі дослідники, що займаються питаннями створення тестових методів. Для кожного завдання встановлюється свій еталон відповіді. При підготовці тестів досягнень застосовується експертна оцінка знань і умінь, що здійснюється паралельно з тестуванням. Спочатку випробовувані виконують тест, і після цього їх опитує групу експертів. Бали по тестах зіставляють з оцінками. Відсоток збігів повинен бути високий - 85-90%, це є гарантією того, що в тест включений основний матеріал по предмету, як його виявили експерти/

Залежно від характеру діяльності випробуваного при виконанні педагогічного тесту розрізняють:

- Тести з вільно конструюються відповідями;
- Тести на додаток відповідей, наведених в завданнях;
- Тести вибору вірних відповідей із загального числа запропонованих в завданні (альтернативні тести);

комбіновані тести.

Для оцінки дедактичних можливостей перерахованих тестів необхідно

зупинитися на особливостях тесту взагалі і окремих видів.

Кожне із завдань тесту, як уже згадувалося раніше, містить інформацію до вирішення. Характер і обсяг її може варіюватися як дидактична допомога випробуваному. Її надлишок може грати позитивну роль в тестах, які використовуються в режимі навчання і навіть діагностики, але виявитися небажаним в тестах для оцінки досягнень, де основною функцією тесту є контроль.

З усіх видів тести вибору мають перевагу в наданні випробуваним свободи у відповідях на завдання, як це буває при самостійному конструюванні відповідей. Це досягається шляхом включення в число відповідей до завдань альтернативного варіанту - «відповідь інший», маючи на увазі відповідь, відмінний від усіх інших, наведених до завдання.

Від виду тесту найістотнішим чином залежить, наскільки проста і технологічна обробка результатів тестування. У цьому сенсі поза конкуренцією тести вибору. Тоді як при вільному конструюванні відповідей випробуваними виявляється виключно складно обробляти результати тестування .

Закінчуючи розгляд видів тесту, слід згадати про так званих ситуаційних тестах, в яких всі завдання ставляться до опису однієї і тієї ж ситуації. До ситуаційним тестів ставляться тести у формі програм вирішення завдань і алгоритмів певного типу. У них послідовність завдань диктується логікою вирішення розглянутих Задачного ситуацій. Результати виконання одних завдань можуть бути використані при виконанні наступних.

Як було зазначено вище, найбільш широке поширення в педагогічному тестуванні отримали тести вибору. Порівнюючи їх з іншими видами педагогічних тестів, слід зазначити простоту обробки результатів їх виконання, що в рамках масового навчання має особливу цінність.

Зустрічається думка про те, що ознайомлення випробовуваних з помилковими відповідями небажано, що ці помилки можуть закріпитися в їх пам'яті, не має сенсу.

Навпаки, робота з такими відповідями дозволяє розкрити помилкові думки і їх зруйнувати.

Відповіді до завдань тесту можуть перебувати в різних відносинах один до одного: протиставлення, протилежності, коли вибір одного з них в якості вірного є твердженням помилковості всіх інших; рівноправності, коли вірними можуть виявитися кілька із запропонованих відповідей; доповнення один одного, коли тільки кілька відібраних відповідей в сумі дають закінчений відповідь на поставлене в завданні питання.

Процес складання педагогічних тестів є справою виключно відповідальним і трудомістким. Оскільки вони є основою контролю, діагностики в управлінні процесом навчання, то стають зрозумілими виключно високі вимоги до змісту і форми тестів. Якщо сказати в найзагальнішому вигляді, то тест повинен бути придатним (дійсним) з точки зору поставлених перед ним цілей вимірювання. Поняття валідності (переклад слова з англійського valid - придатний, придатний) трактується досить широко.

В теорії тестування розрізняють два типи валідності: зовнішню і внутрішню. Зовнішня валідність властива тестів, які не мають подібних компонентів з оцінюваної системою. Внутрішня валідність властива тим тестів, які містять в собі компоненти оцінюваної педагогічної системи. Міра валідності тесту визначається декількома способами, наприклад, порівнянням з тестом-еталоном і зіставленням з об'єктивнішими показниками, отриманими іншими методами.

Зіставлення з об'єктивними показниками передбачає порівняння досягнень випробуваного в тесті з результатами, отриманими за допомогою експертів. В цьому випадку зазвичай беруться оцінки, що виставляються випробуваному викладачами-експертами традиційним чином, без використання тестів, після чого результати за завданнями тесту і за оцінками експертів корелюються. Якщо отримані однотипні зміни загальних результатів, то вважається, що тест має валідність.

Тест вважається змістовно дійсним, якщо аспекти вимірювання відповідають цілям тестування (аспектно повнота тесту). При цьому основна увага в тесті (найбільше число завдань) приділено найбільш важливим з аспектів вимірювання (тест збалансований). Нарешті, послідовність, в якій розташовані завдання і відповіді до них, логічно обгрунтовані, такий тест являє собою єдине ціле (конструктивно цілісний). У педагогічних тестах, що розробляються для контролю, діагностики знань, валідність за змістом відіграє вирішальну роль .

Вибір критеріїв, за якими належить оцінювати результати роботи випробовуваних з тестом, повинен бути бездоганний (критеріальна валідність), гарантуючи надійність, об'єктивність, забезпечуючи потрібну точність вимірювання, даючи можливість розділити випробовуваних (диференційні можливості тесту). Вважається, що до тестових завдань відносяться ті, які відповідають одночасно трьом вимогам - коректності змісту, формі і наявності системоутворюючих властивостей. Першу вимогу відповідають завдання, питання, і тому ця ознака необхідний, але не достатній. Тестові завдання являють собою не питання і не задачі, а твердження, які в залежності від відповідей піддослідних можуть перетворюватися в справжні або неправдиві висловлювання. При розробці педагогічних тестів дотримуються цього твердження. Звичайні питання не можуть бути помилковими або істинними, а відповіді на них бувають дуже розпливчатими і багатослівними, що для перевірки їх істинності потрібні значні витрати інтелектуальної енергії і праці. Таким чином традиційні питання і їх відповіді не є технологічними, внаслідок чого їх не рекомендується включати в тест.

Завдання в тестовій формі - це завдання, до яких, крім змісту, пред'являються вимоги тестової форми, що робить їх формулювання більш лаконічними. Найжорсткіші вимоги пред'являються мови тесту - однозначність його розуміння випробовуваними, виразність, лаконічність. Таким чином, для розробки і застосування в навчальному процесі розглянутих вище тестів необхідно враховувати

об'єктивні характеристики підготовчого процесу. А для цього потрібно:

1)Виділити мети тестування.

2) Відібрати педагогічні ситуації.

3) Перекласти (змоделювати) педагогічні ситуації на мову навчальних завдань.

4) Переконструювати навчальні завдання в тестові завдання.

5) Вибрати й оцінити еталони відповідей.

6) Розробити план тесту. (Передбачається розкладка необхідного числа завдань, виходячи із загального граничного числа завдань різного рівня складності).

7) Виділити аспекти тестування.

8) Скласти тест. (Ця робота повинна проводитися досвідченими викладачами).

9) Розташувати завдання в тесті з урахуванням системоутворюючих зв'язків.

Слід зазначити, що педагогічний тест не може складатися з тільки легких або важких завдань, він повинен мати завдання різного рівня складності, починаючи з легких і закінчуючи складними, на які можуть правильно відповісти більшість випробуваних в тестовій групі .

10) Перевірити експериментально розроблений тест.

На закінчення слід зазначити, що ефективність педагогічного тесту визначається творчим підходом укладачів до розробки системи завдань, що відповідають конкретним цілям тестування.

### 1.3 Аналіз та огляд існуючіх тестових систем

Аналіз информації в мережі інтернет показав, що зараз найбільш розповсюдженою тестової платформою для тестування студентів є Moodle. Moodle - це навчальна платформа для поєднання студентів, викладачів та адміністраторів в одну систему для создания комфортних умов для дистанційного навчання.Moodle переклад на десятки мов, в тому чиіслі і українською. Система використовується з 2014 року - у 199, у 2019 - 229 странах світу, існує понад 80 тисяч зареєстрованих сайтів які працюють на базі Moodle.

Moodle Включає в себе:

- Завдання
- Навчальні форуми
- Завантаження файлів виконаних завдань
- Чат
- Календар подій
- Новини и анонси подій (для різніх груп: сайт, курс, група, спеціальність)
- Тестування
- Вікіпедія

*Можливості для студентів:*

- Можливість доступу до матеріалів (лекції, Практичні / лабораторні та самостійні роботи;
- Інструменти для праці в групі (Вікіпедія, чат, форуми, семінари, вебінарі);
- Можлівість відсліджування прогресу дистанційного курсу студентом;
- Можлівість перевірки результатів пройдених тестів;
- Можлівість коммунікації з викладачем через форум, повідомлення та чат;
- Можлівість завантаження різноманітних файлів студента зі зробленими завданнями;
- Можлівість нагадувань про різноманітні події в навчальному курсі.

*Можливості для викладачів:*

- Засоби для створення дистанційніх курсів;
- Розміщення навчальних матеріалів (лекції, Практичні / лабораторні та самостійні роботи; А також додаткові матеріалів (книги, довідники, посібники, методичні розробки) у різнзноманітних форматах, а також відео, аудіо та презентаційні матеріали у різніх форматах.
- Швидке редагування навчальних матеріалів.

- Реалізація тестів різного типу;
- Автоматичне формування тестів;
- Автоматизація процессу перевірки знань та звітів відносно проходження студентами курсу та звітів відносно проходження студентами тестів;
- Підключення різних плагінів до курсу дозволяє викладачу використовуваит різноманітні сторонні програмні засоби для розгортання дістанційного навчання.

### 1.4 Дослідження найбільш розповсюдженої тестової системи

З безлічі функцій системи дистанційного навчання Moodle, в рамках цієї роботи, нас цікавлять засоби тестування студентів. Проаналізуємо їх, виділимо плюси та мінуси, які згодом використаємо для створення більш досконалої системи тестування. Портал побудований з добре зарекомендованою собі системою авторизації. Після успішного завершення якої користувачі отримують доступ до ресурсів сайту. Незареєстровані користувачі - повинні пройти реєстрацію на сайті, в ході якої новий користувач вибирає собі логін і пароль для доступу до інформації сайту, або адміністратор має зробити це сам, а потім видати дані для входу користувачам. На рисунку 1.1 зображена авторизація будь якого користувача в системі Moodle.

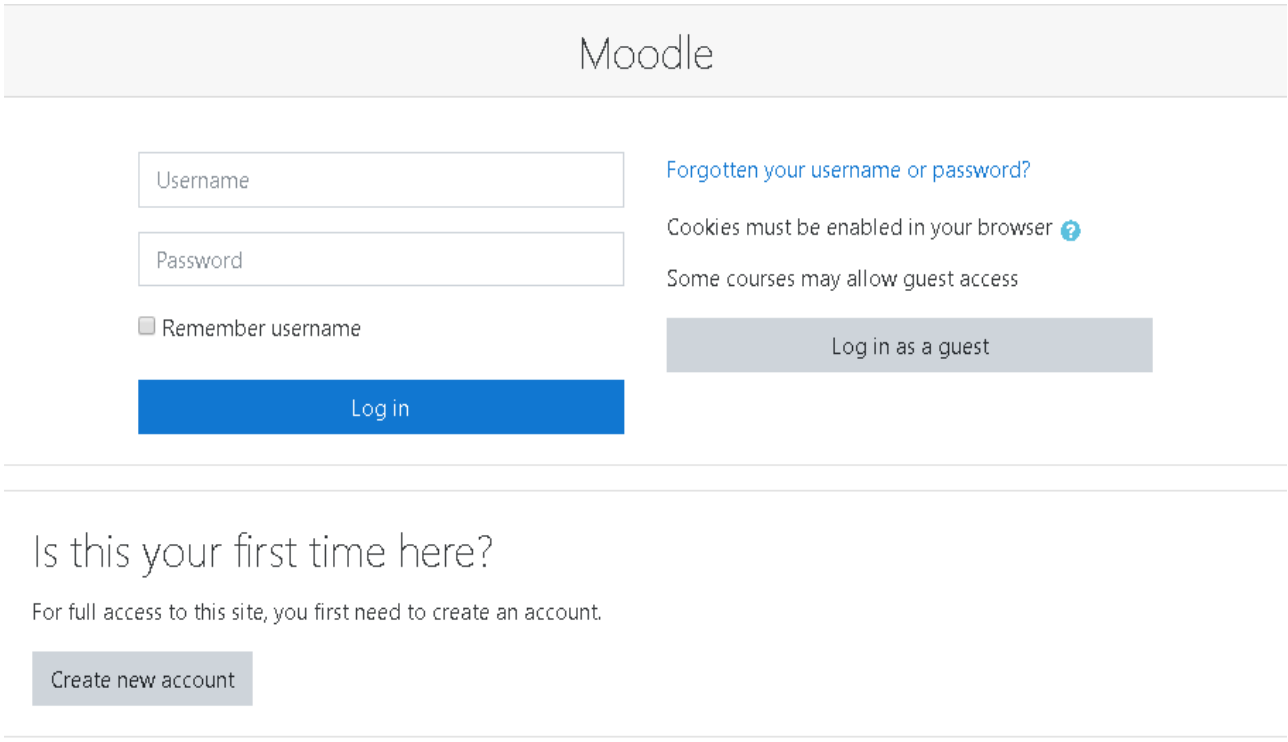



Рисунок 1.1 - Авторизація

Після авторизації потрапляємо на головну сторінку навчального порталу Moodle. Нас цікавлять засоби тестування студентів, тому для початку треба створити хоча б один тест. Не можна просто відкрити Moodle і зібрати тест. Спершу доведеться створити курс, прописати назву і тільки потім сюди можна додати перевірочні завдання. Така у Moodle логіка. Щоб створити курс, перейдемо у вкладку Адміністрування → Курси → Додати курс. Курс в Moodle, як книга. Він складається з окремих глав - розділів, в які додають теорію і практичні завдання. Скільки розділів знадобиться для навчання, зазвичай вибирають заздалегідь. Зараз досить одного, оскільки ми створюємо тільки тест. Поставимо «1» навпроти рядка Кількість розділів. Тепер в курс можна додати тест. Для цього натиснемо “Перейти до курсу” “Додати елементи або ресурс”. На екрані з'явиться меню інструментів Moodle. У розділі “Елементи обміну”, вибираємо тест. Вказуємо період тестування. Спершу виберемо дату і час початку тестування. Наприклад, 17 вересня о 11.00 - в цей час тест автоматично відкриється. До цього моменту студенти в особистому кабінеті

будуть бачити тільки опис майбутнього іспиту.Завершення тестування виставимо на наступний день - рівно в 12.00 тест закриється. Якщо користувач не відповів на питання в термін, він автоматом отримає незалік.Вкажемо початок і закінчення тестування, обмежуємо час тесту. Це залежить від складності перевірочних завдань.

Коли користувач почне відповідати на питання, то побачить таймер зі зворотнім відліком. Він показує, скільки часу залишається на проходження іспиту. Коли час закінчиться, тест автоматично закриється. Інші налаштування в блоці залишаємо за замовчуванням і переходимо до наступного розділу.

При стандартних налаштуваннях студенти мають право відповідати на один і теж питання, поки не набридне. В результаті тест можна пройти методом тику - допустили помилку, відразу поміняли варіант відповіді. І так до нескінченності, поки не набрали максимальний бал. Налаштування кількості спроб можна побачити на рис 1.2

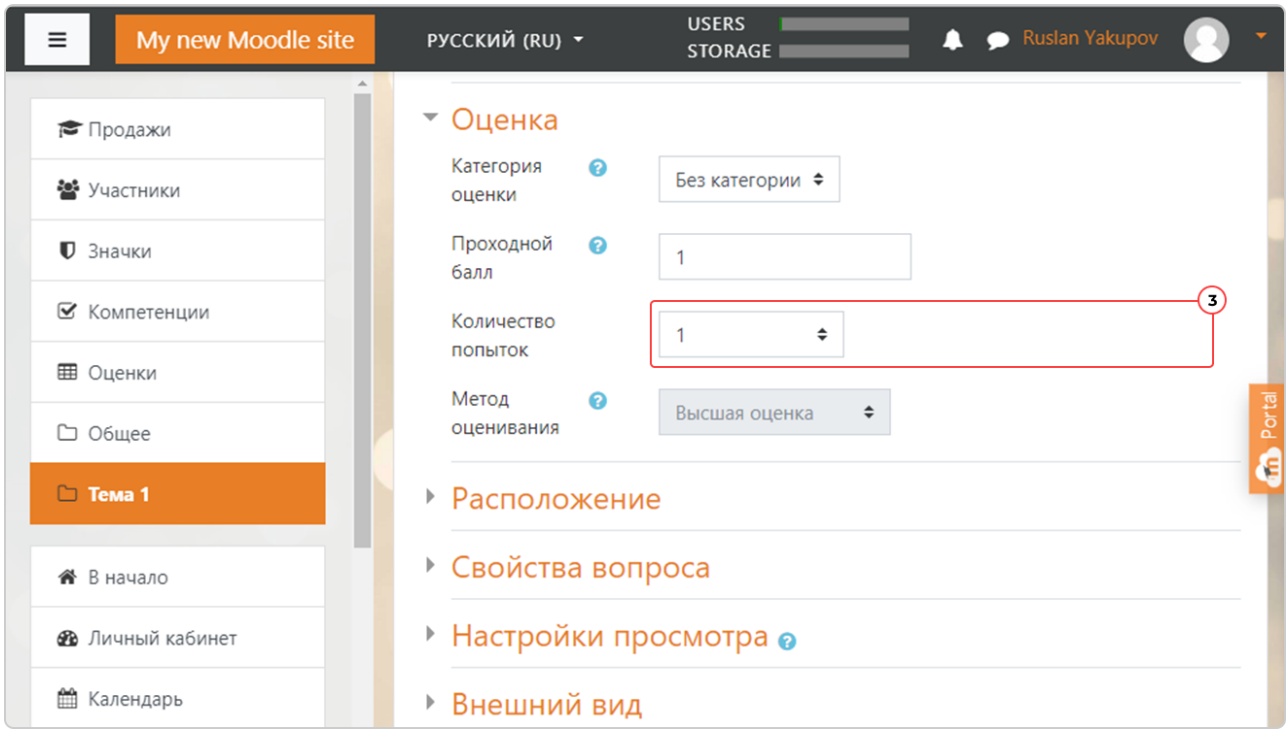



Рисунок 1.2 - Кількість спроб

Далі вказуємо період тестування. Спершу виберемо дату і час початку тестування. Наприклад, 17 листопада 11.00 - в цей час тест автоматично відкриється. До цього моменту студенти в особистому кабінеті будуть бачити тільки опис майбутнього іспиту.Завершення тестування виставимо на наступний опівдні - рівно о 12.00 тест закриється. Якщо користувач не відповів на питання в термін, він автоматом отримає незалік. Налаштування цих параметрів тестування можна побачити на рисунку. 1.3.

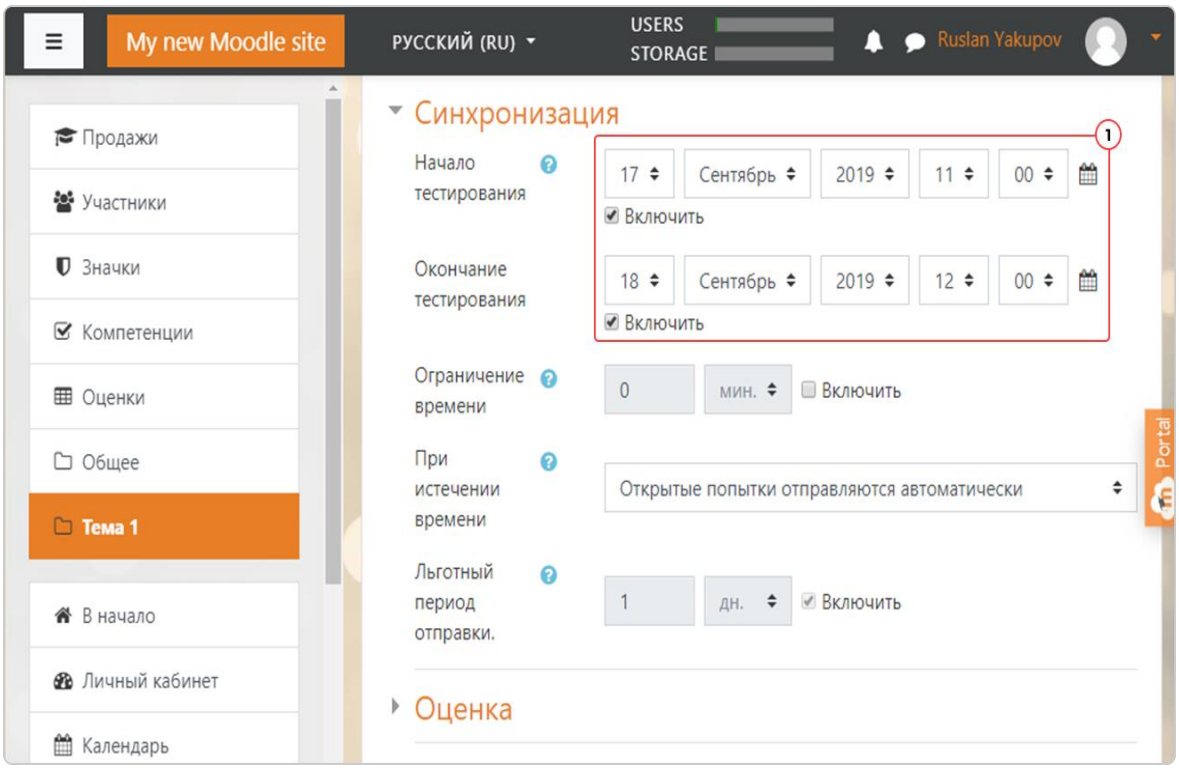



Рисунок 1.3 - Налаштування часу доступності тесту

Також наявна можливість обмеження часу самого тесту. Коли користувач почне відповідати на питання, то побачить таймер зі зворотнім відліком. Він показує, скільки часу залишається на проходження іспиту. Коли час закінчиться, тест автоматично закриється та буде показано результат.Також можливо згрупувати завдання по блокам, помістити кожне питання на окремій сторінці або все на одній (рис 1.4).

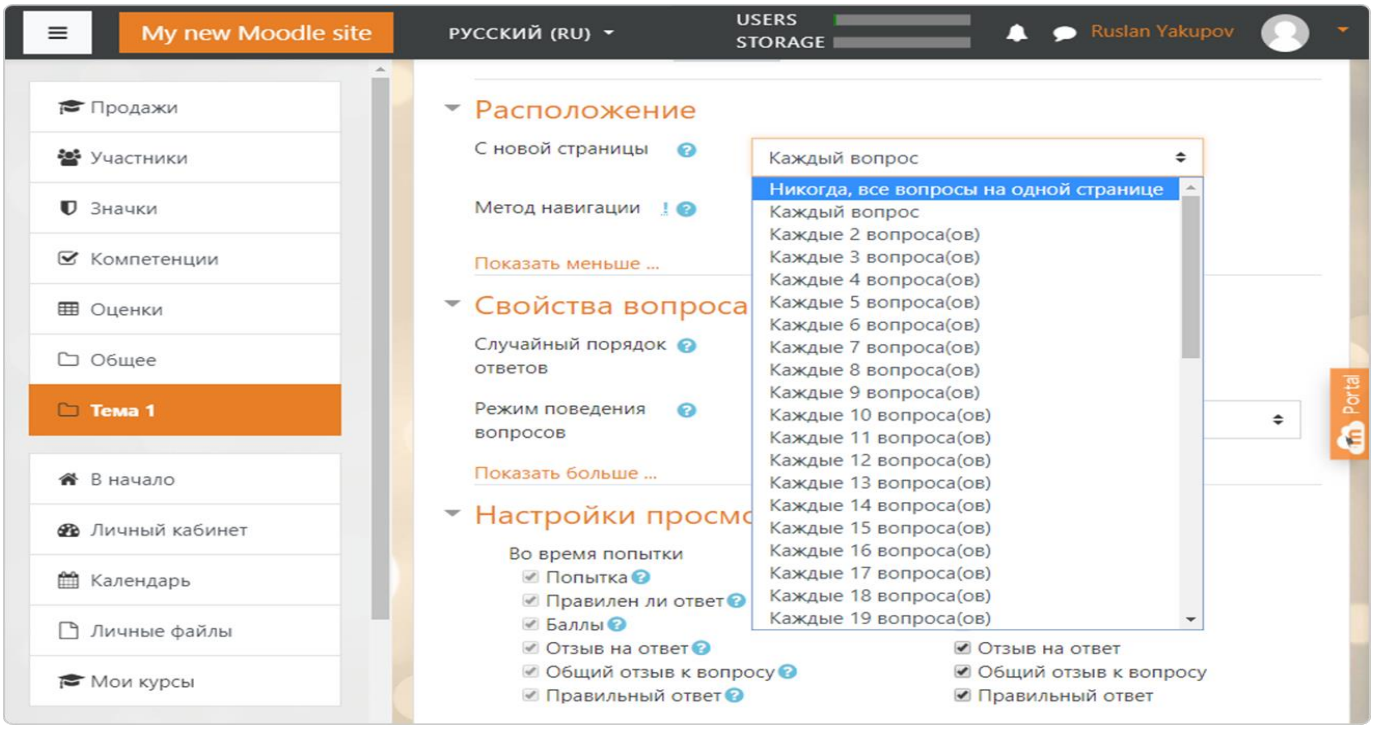



Рисунок 1.4 - Налаштування порядку запитань

Після цього можна вибрати порядок запитань, вони можуть йти по порядку, або з’являтись в випадковому порядку. У другому випадку списати буде складніше. Далі можливо відредагувати підсумковий відгук - повідомлення, яке бачить користувач після тестування. Текст повідомлення змінюється в залежності від набраного бала: відмінників можна похвалити, а двієчникам розповісти про правила перездачі. З основними настройками розібралися, інше налаштування відповідають лише за відображення,їх можна залишити за замовчуванням. Натискаємо: “Зберегти і показати”.З настройккамі тесту закінчили. Наступний етап - додати перевірочні завдання. Система відправить нас на нову сторінку. У Moodle декілька основних типів питань для перевірки знань:

Moodle має декілька видів тестів:

1) В закритій формі (множинний вибір). Завдання складається з декількох можливих варіантів , з яких вірними є одна або їх може бути декілька. Це можна змінити при

створенні питання. Сума вірних відповідей має бути 100%. Такий тип питань , мабуть, є найбільш розповсюдженим.

2) На відповідність. Тут від випробовуваного потребується з'єднати компоненти двох списків, причому у другому переліку едементів може бути більше, ніж в 1. Парні елементи можуть бути вибрані з випадаючого переліку. Розвитком такого варіанту питання є випадкове питання на відповідність - перемішуються варіанти відповідей з декількох питань.

3) Вірно / невірно. Питання типу (так або ні) - самий використовуваний.

4) Коротка відповідь. Потрібно написати слово або декілька слів. При цьому можна зробити враховування регістру символів. Якщо регістр букв не враховується, то слово можна вводити буквами будь якого регістру.

5) Числовий. При відповіді треба ввести число. Це використовується, коли в питанні необхідно щось обчислити, варто зауважити,що відповідь може мати похибку,яка може бути задана викладачем.

6) Вбудована відповідь. Це питання, яке може містити в тексті комбінацію випадаючого списку, поля для введення короткої відповіді і / або числової відповіді. Може мати більшу вагу чим звичайне питання, але більш складна у розробці та вимагає знання коду системи Мудл.

7) Твір. Питання відкритого типу, які вимагають розгорнутої відповіді,наприклад: твори або звіти. Таке питання оцінюється викладачем.

8) Опис. Це, власне, навіть не питання, а деякий текст, наприклад, описує перехід до наступної групи питань.

9) Обчислюваний . Питання, в формулюванні якого можна увімкнути певні шаблони, які кожного разу мають нові значення. Таким чином можливо організувати одержання кожним випробовуваним індивідуального завдання. Щоб освоїти алгоритм роботи над тестом, створимо два найпопулярніших види завдань для тестування: «Правильно / Неправильно» і «Множинний вибір». Щоб додати завдання

на множинний вибір, натисніть Додати → Нове запитання → Множинний вибір → Додати. Пропишемо назву, текст питання і варіанти відповідей з шаблону.

У завданні на множинний вибір може бути один або кілька правильних варіантів відповіді. У нашому прикладі тільки один пошук правильної відповіді. Так і вкажемо в настройках (рис 1.5):

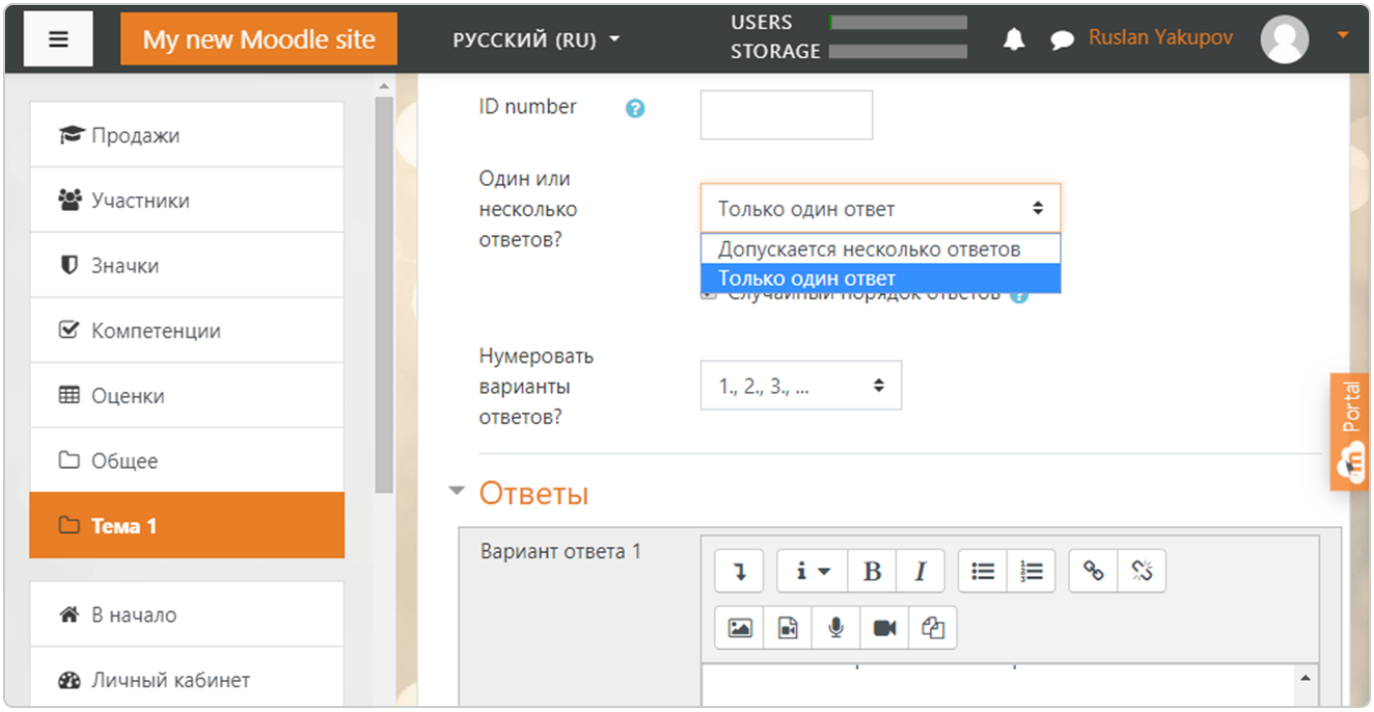



Рисунок 1.5 - Створення завдання типу "множинний вибір"

Тепер виберемо правильну відповідь і виставимо для нього оцінку в 100%. Решта відповідей помилкові - за них не даємо нагороди. Інші налаштування залишимо за замовчуванням,додаємо всі питання та натискаємо “Зберегти”.Готове завдання виглядає так(рис 1.6):

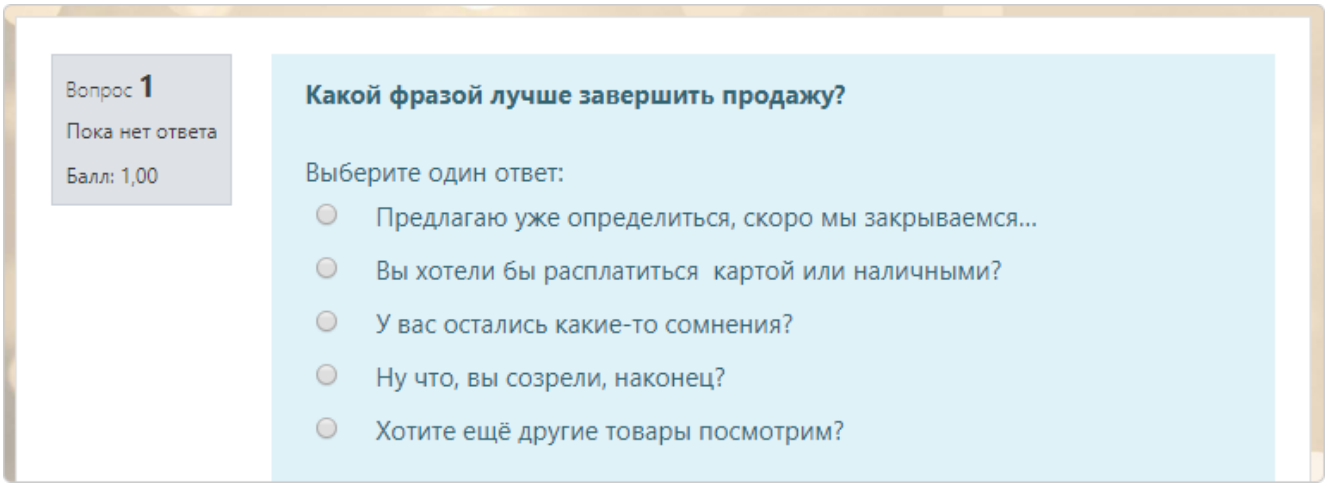



Рисунок 1.6 - Готове завдання типу “множинного вибору”

У вас на вибір є ще кілька налаштувань відображення:

- Правильна чи відповідь - тест покаже помилився користувач або відповів правильно.
- Бали - користувач бачить скільки балів набрав за відповідь
- Загальний відгук до питання - зворотний зв'язок для користувача після відповіді.

Щоб активувати ці настройки, натисніть Попередній перегляд і виберіть в новому вікні потрібні опції. Додамо ще декілька завдань, налаштуємо опції відображення. Після цього тест вважається готовим.

Банк питань - загальний склад для перевірочних завдань Moodle. Тут зберігаються всі питання, які ви коли-небудь створили. Якщо вам потрібно зібрати тест для нового курсу, не потрібно робити все з нуля - ви просто копіюєте заготовки з банку .. Банк питань економить час.Завдання, які ми зібрали сьогодні, вже в банку питань. Щоб їх подивитися, перейдемо в курс через особистий кабінет. У списку виберіть розділ “Банк питань”(рис1.7). Тут ви можете створити нові завдання або подивитися старі.

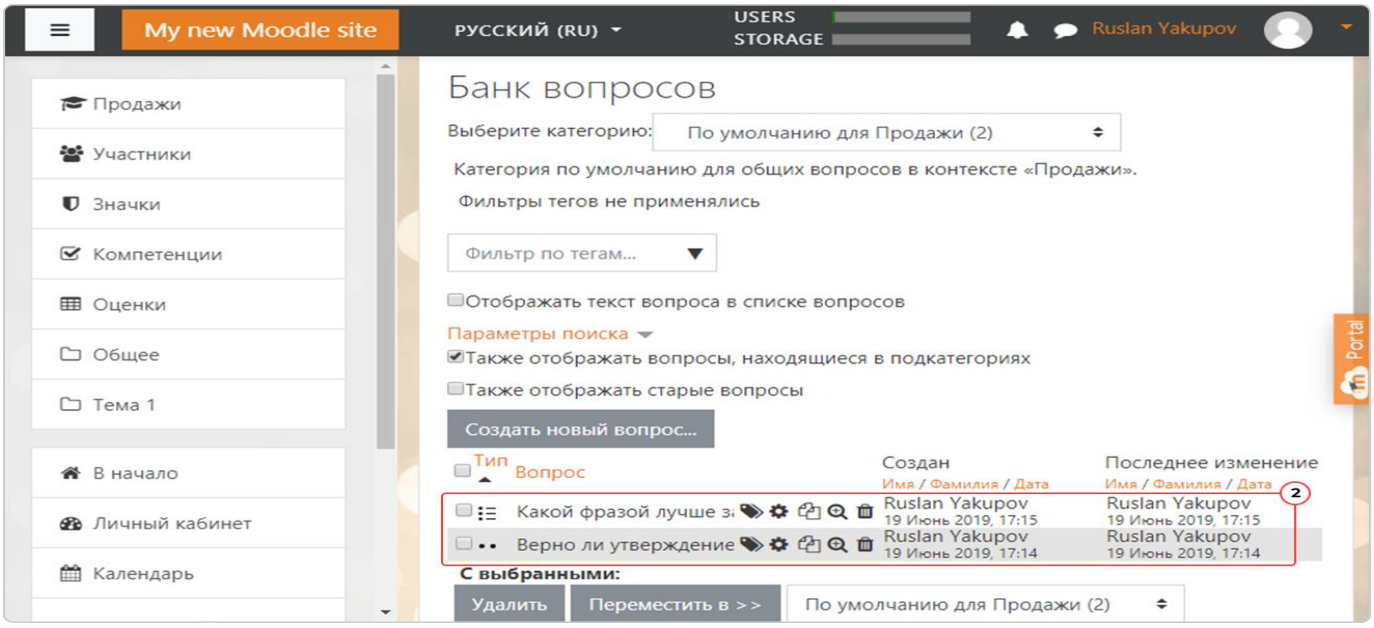



Рис 1.7 - Банк питань

Наступного разу ви можете задати всі завдання тут і зібрати з них новий тест.У Moodle широкий набір можливостей для тесту, але через велику кількість налаштувань інтерфейс виглядає перевантаженим. Новачкові складно відшукати потрібну опцію, не заглядаючи в документацію. До принципу роботи потрібно теж звикнути: щоб створити тест, доводиться окремо налаштовувати курс, задавати параметри тестування і кожного завдання. В результаті робота затягується. Вирішити завдання швидше допоможуть конструктори типу iSpring Suite. Ви збираєте інтерактивний тест з готових шаблонів в одному вікні, не перемикаючись між численними сторінками. Готовий тест сумісний з Moodle і підлаштовується під будь-який розмір екрану. Але для цього потрібно купити, завантажити та встановити додаткове ПО, що, звичайно, не завжди зручно.

Після виконаного аналізу засобів тестування студентів системою Moodle можемо виділити її плюси та мінуси, які при створенні нової системи тестування ми врахуємо

Плюси :

1. Встановлення системи Moodle досить просте та займає мало часу
2. Тестові засоби в Moodle інтегровані в навчальні курси, тому для дистанційного

навчання зручніше мати все в одному місці.

3. Великий вибір типів завдань для тестування.

4. Можливість контролювати дату здачі тесту.

5. Широкі можливості налаштування відображення різних елементів для користувача.

6. Можливість завантажувати тести створені в сторонньому ПО (конструкторах)

7. “Банк питань”, всі питання зберігаються в банку, після чого їх можна використовувати в інших тестах.

# 2 РОЗРОБКА СИСТЕМИ ТЕСТУВАННЯ

## 2.1 Постановка задачі

Необхідно розробити WEB-систему тестування студентів з наступними можливостями:

- Наявність блоку реєстрації та авторізації. З повідомленням про проходження реєстрації на електронну пошту, шифрування пароля за алгоритмом md5, а так само з перевіркою «анти-бот» и можлівістю змінюваті свои персональні дані.

- Категоріювання сторінок сайту за різнімі термінамі. З можлівістю для викладачів создать свои категорії сортуваннях и включать в них Терміни.

- Створення тесту. З можливіть: сортуваннях вопросам у Випадкове порядку за Бажанов викладача, відображення результатів и правильних Відповідей, установки кількості Спроба на проходження тесту, Вибори налаштування по проходження тесту, настройки доступності тестів по числу и годині, установки порога успішної здачі тесту в процентах, Виведення ОЦІНКИ по Завершення тестування (розрахунок ОЦІНКИ відбувається в залежності від набраних відсотка).

- Створення питання з Вибори кількості правильних варіантів ВІДПОВІДІ. З можлівістю размещения Відповідей у Випадкове порядку и з можлівістю Додавання даного питання в уже існуючій тест.

- Створення питання з рукописним відповіддю. Викладач сам перевіряє відповідь студента.

- Створення питання з можлівістю Встановити відповідність между пунктами підпітання, а также создания питання з можлівістю Встановити порядок підпітань.
- Використання групової політики доступу для забезпечення безпеки. Студент, незареєстрованій користувач, адміністратор и викладач.

### 2.2 Серверні засоби для реализации системи тестування

Для розгорнення системи тестування студентів ми будемо використовуват вже перевірене часом рішення, а саме - LAMP.

LAMP (Linux, Apache, MySQL, PHP) - це набір програмного забезпечення з відкритим вихідним кодом, які зазвичай встановлюються на сервер для відображення дінамічніх веб-сайтів и веб-додатків. Ця абревіатура означає операційну систему Linux з встановленим веб-сервером Apache, дані сайту зберігаються в базі Даних MySQL, а динамічний контент обробляється за допомогою PHP.

### 2.3 Установка Apache и настройка файрвол

Веб-сервер Apache є одним з найпопулярнішіх веб-серверів в світі. Він добре задокументований і широко використовується з моменту створення мережі інтернет, що робить його прекрасним вибором для хостингу веб-сайту.

Встановимо Apache використовуючі менеджер пакетів Ubuntu apt:

*sudo apt update*

*sudo apt install apache2*

Оскільки ми використовуємо команду sudo, ця команда буде виконана з привілегіями root. Після цього вводимо пароль, пакет apt повідомить, які пакети будут встановлені і скільки місця вони займуть на диску.

### 2.4 Налаштування файрволу для контролю веб-трафіку

Тепер переконаємося, що наш файрвол пропускає HTTP и HTTPS трафік.. Для початку переконаємося, что система має профіль для Apache за допомогою такої команди:

*sudo ufw app list*

Після цього перевіримо налаштування профілю Apache Full, він повинен дозволяти трафік для портів 80 и 443:

*sudo ufw app info "Apache Full"*

Наступним кроком дозволимо HTTP и HTTPS трафік для цього профілю:

*sudo ufw allow in "Apache Full"*

Перевірити результат установки можна набравши в веб-браузері IP адресу серверу, в нашому випадку: "*127.0.0.1*".

Після введення ip адреси бачимо сторінку Apache, що іідображається за замовчуванням для інформації і цілей тестування.Це означає, що встановлення пройшло успішно. Сторінка виглядає таким чином:

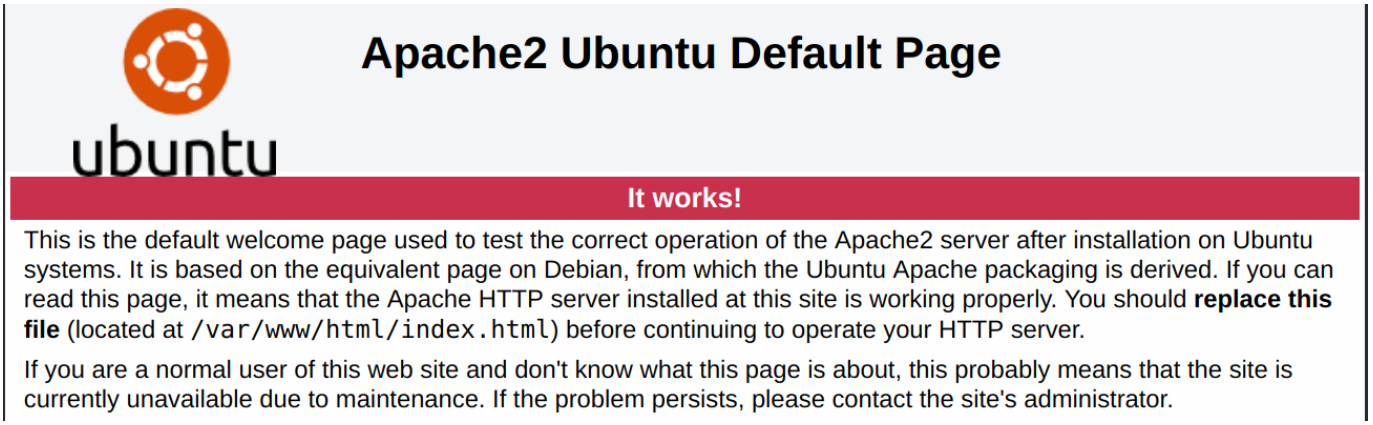



Рисунок 2.1 - Сторінка Apache за замовчуванням

### 2.5 Налаштування MySQL

Тепер, коли наш веб-сервер встановлений и запущений, треба встановити MySQL. MySQL це система управління базами даних. Вона організовує і забезпечує доступ до баз даних, в яких наша система может зберігати інформацію.

Ми вікористуємо apt для завантаження і установки програмного забезпечення:

*sudo apt install mysql-server*

Після завершення встановлення нам треба виконати деякі додаткові команди, щоб наше оточення MySQL було налаштовано безпечним чином. введемо наступну команду:

*sudo mysql_secure_installation*

Далі буде запропоновано задати пароль для користувача root для MySQL. Це адміністративний акаунт користувача в MySQL, який має підвіщені привілегії. Ми можемо розглядаті його, як аналог користувача root для самого сервера (з тією лишь різніцею, що це аккаунт для MySQL). Задаємо сильний унікальний пароль.

На всі следующие питання просто вводимо Y та натискаємо клавішу ENTER для вибору налаштуваннь за замовчуванням.

### 2.6 Налаштування PHP

PHP це компонент, який буде обробляти код для відображення дінамічного контенту. Він может запускати скрипти, підключатися до наших баз Даних MySQL, для отримання інформації, та передавати оброблений контент в наш веб-сервер, для відображення.Ми також додамо деякі допоміжні пакети, щоб код на PHP міг працювати з нашим сервером Apache, а також звертатися до бази даних MySQL:

*sudo apt install php libapache2-mod-php php-mysql*

Дана команда повинна встановити PHP без будь-яких проблем. Незабаром ми це перевірімо.У більшості випадків, ми захочемо змінити способ, який Apache вікорістовує для пошуку файлів при запиті директорії. Зараз, якщо користувач запитує діректорію з серверу, Apache ,в першу чергу, буде шукати файл з назвою "*index.htm;"*. Ми хочемо, щоб наш веб-сервер віддавав перевагу PHP файлам, так що ми налаштуємо Apache таким чином, щоб спочатку  він шукав файли типу: *"index.php"*.

Для цього введемо наступну команду, щоб відкрити файл "*dir.conf"* в текстовому редакторі з прівілеями користувача root:

*sudo nano /etc/apache2/mods-enabled/dir.conf*

Файл буде виглядати наступним чином:

*/etc/apache2/mods-enabled/dir.conf*

*<IfModule mod_dir.c>*

*DirectoryIndex index.html index.cgi index.pl index.php index.xhtml index.htm*

*</ IfModule>*

Ми перемістимо індексний файл “index.php”, виділений вище, на перше місце після директорії *“DirectoryIndex*” наступним чином:

*/etc/apache2/mods-enabled/dir.conf*

*<IfModule mod_dir.c>*

*DirectoryIndex index.php index.html index.cgi index.pl index.xhtml index.htm*

*</ IfModule>*

Після внесення правок збережемо та закриємо файл, натіснувши сполучення клавіш CTRL-X.Після цього нам необходимо перезапустити Apache дляз астосування внесених змін. Робимо це за помощью команди:

*sudo systemctl restart apache2*

Тепер наш LAMP встановлений та налаштований. Однак, перед внесенням будь-яких других змін і перед установкою додатків нам треба протестувати PHP на випадок можливих проблем.

### 2.7 Тестування роботи PHP

Щоб перевірити, що наша система налаштована належним чином, ми створимо простий PHP скрипт. Назвемо його info.php. Щоб Apache знайшов и обробив файл коректно, він винен бути збережений в спеціальній діректорії, яка називається "web root" .В Ubuntu 18.04 дана діректорія находится за адресою / var / www / html /. Ми можемо створити там файл ввівши таку команду:

*sudo nano /var/www/html/info.php*

Відкріється порожній файл. Введемо в файл наступний текст, який буде коректним PHP кодом:

*info.php*

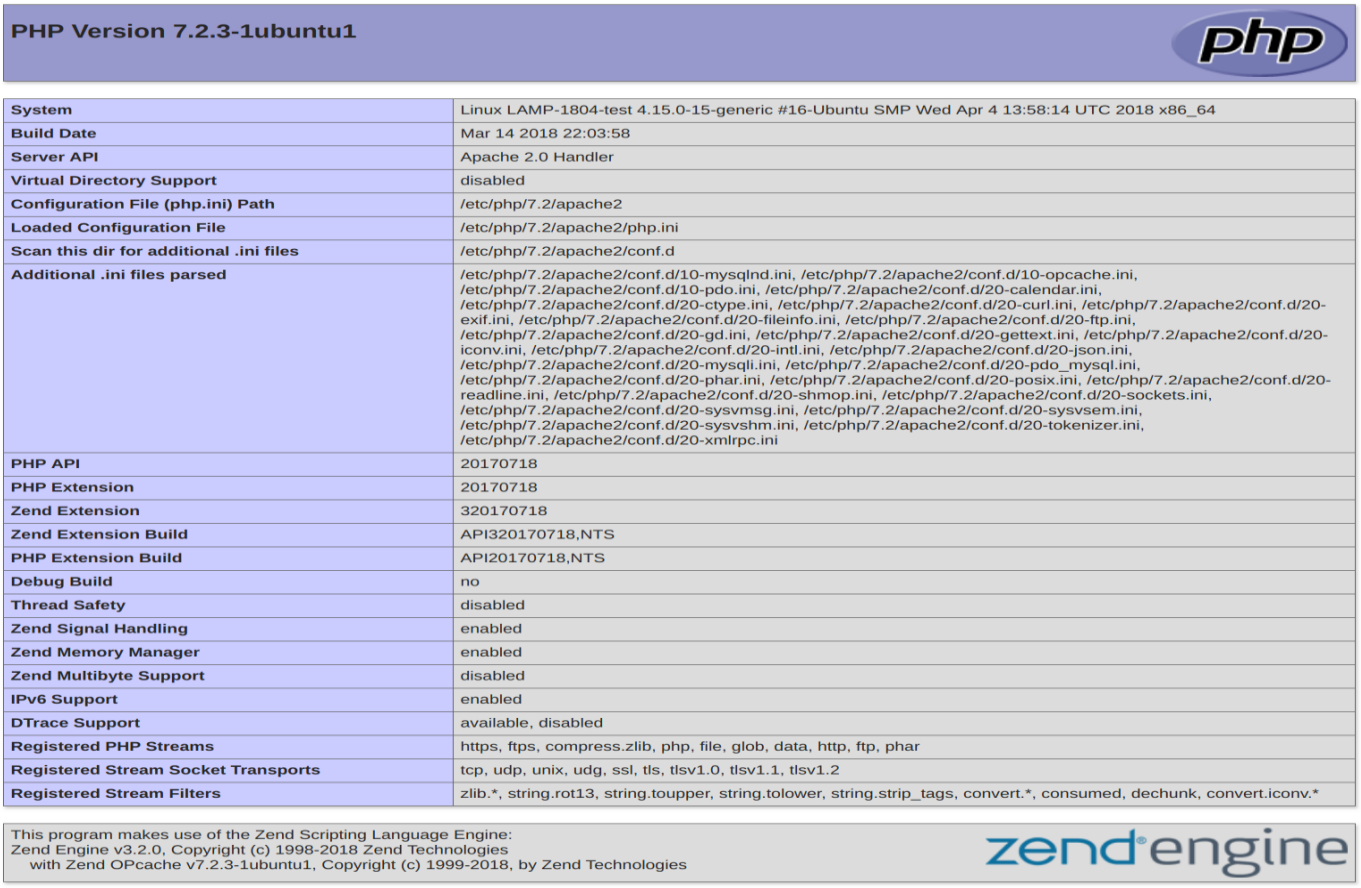
*<? Php*

*phpinfo ();*

*?>*

Після весення змін збережемо і закриємо файл.Тепер ми можемо перевірити, чи веб-сервер коректно відображає контент, згенерований PHP скриптом. Для перевірки нам треба відкрити цю сторінку в веб-браузері. Нам знову нужно перейти по айпі адресі нашого сервера.

Відкріємо адресою: *"127.0.0.1/info.php"*. З'явилась наступна сторінка:



**PHP Version 7.2.3-1ubuntu1**

| | |
|---|---|
| **System** | Linux LAMP-1804-test 4.15.0-15-generic #16-Ubuntu SMP Wed Apr 4 13:58:14 UTC 2018 x86_64 |
| **Build Date** | Mar 14 2018 22:03:58 |
| **Server API** | Apache 2.0 Handler |
| **Virtual Directory Support** | disabled |
| **Configuration File (php.ini) Path** | /etc/php/7.2/apache2 |
| **Loaded Configuration File** | /etc/php/7.2/apache2/php.ini |
| **Scan this dir for additional .ini files** | /etc/php/7.2/apache2/conf.d |
| **Additional .ini files parsed** | /etc/php/7.2/apache2/conf.d/10-mysqlnd.ini, /etc/php/7.2/apache2/conf.d/10-opcache.ini, /etc/php/7.2/apache2/conf.d/10-pdo.ini, /etc/php/7.2/apache2/conf.d/20-calendar.ini, /etc/php/7.2/apache2/conf.d/20-ctype.ini, /etc/php/7.2/apache2/conf.d/20-curl.ini, /etc/php/7.2/apache2/conf.d/20-exif.ini, /etc/php/7.2/apache2/conf.d/20-fileinfo.ini, /etc/php/7.2/apache2/conf.d/20-ftp.ini, /etc/php/7.2/apache2/conf.d/20-gd.ini, /etc/php/7.2/apache2/conf.d/20-gettext.ini, /etc/php/7.2/apache2/conf.d/20-iconv.ini, /etc/php/7.2/apache2/conf.d/20-intl.ini, /etc/php/7.2/apache2/conf.d/20-json.ini, /etc/php/7.2/apache2/conf.d/20-mysqli.ini, /etc/php/7.2/apache2/conf.d/20-pdo_mysql.ini, /etc/php/7.2/apache2/conf.d/20-phar.ini, /etc/php/7.2/apache2/conf.d/20-posix.ini, /etc/php/7.2/apache2/conf.d/20-readline.ini, /etc/php/7.2/apache2/conf.d/20-shmop.ini, /etc/php/7.2/apache2/conf.d/20-sockets.ini, /etc/php/7.2/apache2/conf.d/20-sysvmsg.ini, /etc/php/7.2/apache2/conf.d/20-sysvsem.ini, /etc/php/7.2/apache2/conf.d/20-sysvshm.ini, /etc/php/7.2/apache2/conf.d/20-tokenizer.ini, /etc/php/7.2/apache2/conf.d/20-xmlrpc.ini |
| **PHP API** | 20170718 |
| **PHP Extension** | 20170718 |
| **Zend Extension** | 320170718 |
| **Zend Extension Build** | API320170718,NTS |
| **PHP Extension Build** | API20170718,NTS |
| **Debug Build** | no |
| **Thread Safety** | disabled |
| **Zend Signal Handling** | enabled |
| **Zend Memory Manager** | enabled |
| **Zend Multibyte Support** | disabled |
| **IPv6 Support** | enabled |
| **DTrace Support** | available, disabled |
| **Registered PHP Streams** | https, ftps, compress.zlib, php, file, glob, data, http, ftp, phar |
| **Registered Stream Socket Transports** | tcp, udp, unix, udg, ssl, tls, tlsv1.0, tlsv1.1, tlsv1.2 |
| **Registered Stream Filters** | zlib.*, string.rot13, string.toupper, string.tolower, string.strip_tags, convert.*, consumed, dechunk, convert.iconv.* |

This program makes use of the Zend Scripting Language Engine:
Zend Engine v3.2.0, Copyright (c) 1998-2018 Zend Technologies
with Zend OPcache v7.2.3-1ubuntu1, Copyright (c) 1999-2018, by Zend Technologies

Рисунок 2.2 - Сторінка info.php

Дана сторінка містіть інформацію про ваш сервер з точки зору PHP. Вона корисна для налагодження та щоб упевнитися в коректності застосування налаштувань.Якщо все пройшло успішно, значить наш PHP працює коректно.

### 2.8 Структура тестового порталу

Розглянемо детальніше складові тестового порталу:

База даних (БД) порталу є сховищем всієї інформації порталу, доступ до якої здійснюється через програмне ядро. БД порталу є реляційної базою даних - найбільш

придатною і здатної задовольнити всі завдання при написанні подібного роду програмних продуктів. Відповідно до правил реляційних БД база даних ділиться на таблиці, зокрема і цей портал. Тестовий портал складається з наступних таблиць:

- курси (kurs)
- користувачі (users)
- доступ (dostup)
- ліміти на перездачу (limits)
- ліміти на кількість питань (ctrl_counts)
- питання (questions)
- студенти (students)
- результати (results)

**Таблиця «Курси»**. Ця таблиця буде містити в собі інформацію про курси порталу. У таблиці 1 наведена структура цієї таблиці 2.1:

Таблиця 2.1 – Інформація про курси порталу

| № | Назва поля | Тип поля | опис |
|---|---|---|---|
| 1 | kurs_ID | Лічильник | Числовий ідентифікатор курсу (первинний ключ) |
| 2 | name | текстовий | Назва курсу |
| 3 | Ped | текстовий | П.І.Б тьютора |
| 4 | Act | логічний | активність курсу |
| 5 | Dat | Дата час | Дата останнього редагування |

Поле «kurs_id» є первинним ключем таблиці «Курси» і є сполучною полем з іншими таблицями. Це поле має тип «Лічильник» - це означає, що при вставці нового запису, полі прийме чергове значення віртуального лічильника, який просто інкрементує своє значення на одиницю.

Хотілося б звернути увагу на поле «АСТ» - за допомогою цього поля регулюється активність того чи іншого курсу. За замовчуванням курс завжди активний, приймає значення True в поле «Act». Що розуміється під активністю курсу?

Все просто: якщо курс активний, то студенти мають право доступу до курсу, в іншому випадку - ні. Для чого це потрібно? Припустимо, було виявлено що в тому чи іншому курсі при введенні питань був допущений ряд істотних помилок (питання занадто легкі, невірні відповіді, повтор питань, і т.п.), тоді, щоб уникнути непорозумінь адміністратору порталу буде достатньо всього лише привести курс у неактивний стан до усунення помилок.

**Таблиця «Користувачі»**. Дана таблиця містить інформацію про користувачів тестового порталу, тобто про людей мають безпосередній доступ до порталу. Сюди в першу чергу відноситься адміністратор, а потім оператори. У таблиці 2.2 наведена структура цієї таблиці:

Таблиця 2.2 – інформація про користувачів

| № | Назва поля | Тип поля | опис |
|---|---|---|---|
| 1 | us_ID | Лічильник | Числовий ідентифікатор користувача (первинний ключ) |
| 2 | Fio | текстовий | П.І.Б. користувача |
| 3 | Mail | текстовий | Адреса електронної пошти користувача |
| 4 | Login | текстовий | Логін користувача |
| 5 | Pass | текстовий | пароль користувача |
| 6 | Status | безліч | статус користувача |
| 7 | Dat | Дата час | Дата останнього редагування |

З полями даної таблиці все зрозуміло, назви полів говорять самі за себе. Хіба що поле «Status» вимагає деякого пояснення. Це поле приймає одне з таких: або 'O', або 'A', тобто або оператор, або адміністратор відповідно. По суті це поле в даному випадку можна було замінити на логічне, але перевага безлічі в тому, що сюди можна додати ще десятки значень, тим самим при розширенні і апгрейд тестового порталу не буде проблем з додаванням нових категорій користувачів.Поле «Pass» зберігає в собі паролі користувачів в зашифрованому вигляді, що підвищує безпеку порталу. Здійснення шифрування відбувається на рівні СУБД MySQL за допомогою вбудованої функції *“SHA1*”, яка шифрує текстову інформацію однойменним

криптографічним алгоритмом. Звичайно можна було б використовувати зовнішнє шифрування або власний алгоритм, але краще буде віддати перевагу вже добре зарекомендувавшому алгоритму, до того ж взлом шифру даного алгоритму дуже і дуже непростий.

**Таблиця «Доступ»**. У цій таблиці буде розташовуватися інформація про доступ користувачів (операторів) до тих чи інших курсів. Природно це стосується тільки операторів, тому що адміністратор має повний доступ до всієї інформації порталу. У таблиці 2.3 наведена структура цієї таблиці:

Таблиця 2.3 – інформація про доступ користувачів

| № | Назва поля | Тип поля | опис |
|---|---|---|---|
| 1 | us_ID | Лічильник | Числовий ідентифікатор користувача (зовнішній ключ) |
| 2 | kurs_ID | Лічильник | Числовий ідентифікатор курсу (зовнішній ключ) |
| 3 | Sel | логічний | Дозвіл на читання |
| 4 | Edit | логічний | Дозвіл на редагування |
| 5 | Ins | логічний | Дозвіл на вставку |
| 6 | Del | логічний | Дозвіл на видалення |
| 7 | Lim | логічний | Дозвіл на зміну лімітів на перездачі |
| 8 | Dat | Дата час | Дата останнього редагування |

Поля являють собою зовнішні ключі (us_ID, kurs_ID), які можуть містити в собі тільки значення з певних полів відповідних таблиць. В даному випадку поле «us_ID» містить в собі значення поля «us_ID» з таблиці «Користувачі», а поле «kurs_ID» - значення поля «kurs_ID» таблиці «Курси». Таким чином якщо спробувати вставити в поле «kurs_ID» значення, якого не існує в однойменному полі таблиці «Курси» ми отримаємо помилку про невідповідність зовнішнього ключа і вставка (зміна) буде скасована, тобто проблема невідповідності даних при використанні зовнішніх ключів вирішується на рівні СУБД.

Поля "*sel", "edit", "ins" і "del"* зберігають в собі дозвіл на читання, редагування, вставку і видалення питань того чи іншого курсу, і призначають доступ оператору відповідно до його ідентифікатору. Поле *"lim"* дає або забороняє

оператору можливість редагування лімітів на перездачу контролів (докладніше про контролі дивіться нижче).

**Таблиця «Ліміти на перездачу»**. Дана таблиця зберігає інформацію про ліміти (обмеження) на перездачу контролів, тобто є можливість обмежити кількість перездач того чи іншого контролю. У тому випадку, коли студент досяг ліміту на перездачу даного контролю (наприклад, ліміт контролю дорівнює 3 - студент перездавав 3 рази), подальша перездача буде недоступна. Таким чином у учнів буде стимул для більш глибокого і кращого засвоєння матеріалу обраного курсу. У таблиці 2.4 наведена її структура:

Таблиця 2.4 – ліміти на перездачу

| № | Назва поля | Тип поля | опис |
|---|---|---|---|
| 1 | kurs_ID | Лічильник | Числовий ідентифікатор курсу (зовнішній ключ) |
| 2 | Tek1 | числовий | Перездачі 1 поточного контролю |
| 3 | Tek2 | числовий | Перездачі 2 поточного контролю |
| 4 | Tek3 | числовий | Перездачі 3 поточного контролю |
| 5 | Tek4 | числовий | Перездачі 4 поточного контролю |
| 6 | Prm1 | числовий | Перездачі 1 проміжного контролю |
| 7 | Prm2 | числовий | Перездачі 2 проміжного контролю |
| 8 | Itg | числовий | Перездачі підсумкового контролю |
| 9 | Dat | Дата час | Дата останнього редагування |

Чому обрано саме така система контролю? Звичайно можна було зробити портал заснований на лекційних контролях, тобто після кожної лекції відповідний контроль. Але так як в даному випадку ми маємо справу лише з тестовим порталом, що не містить у собі безпосередньо лекцій, то найкращим варіантом буде саме обраний варіант.

Як і в попередній таблиці встановлені ліміти прив'язуються до того чи іншого курсу, відповідно до його ідентифікатором.

**Таблиця «Ліміти на кількість питань»**. Дана таблиця зберігає інформацію про ліміти (обмеження) на кількість питань того чи іншого контролю при

проходженні його студентами. У таблиці 2.5 наведена структура цієї таблиці:

Таблиця 2.5 – ліміти на кількість питань

| № | Назва поля | Тип поля | опис |
|---|---|---|---|
| 1 | kurs_ID | Лічильник | Числовий ідентифікатор курсу (зовнішній ключ) |
| 2 | Tek1 | числовий | Кількість питань 1 поточного контролю |
| 3 | Tek2 | числовий | Кількість питань 2 поточного контролю |
| 4 | Tek3 | числовий | Кількість питань 3 поточного контролю |
| 5 | Tek4 | числовий | Кількість питань 4 поточного контролю |
| 6 | Prm1 | числовий | Кількість питань 1 проміжного контролю |
| 7 | Prm2 | числовий | Кількість питань 2 проміжного контролю |
| 8 | Itg | числовий | Кількість питань підсумкового контролю |
| 9 | Dat | Дата час | Дата останнього редагування |

Наприклад, якщо проставити в поле *"Тек1"* того чи іншого курсу «5», то при проходженні першого поточного контролю студент повинен буде відповісти на п'ять питань.

**Таблиця «Питання»**. У цій таблиці будуть зберігатися питання всіх курсів і контролів порталу. У таблиці 2.6 наведено структуру цієї таблиці:

Таблиця 2.6 – питання

| № | Назва поля | Тип поля | опис |
|---|---|---|---|
| 1 | q_ID | Лічильник | Числовий ідентифікатор питання (первинний ключ) |
| 2 | kurs_ID | Лічильник | Числовий ідентифікатор курсу (зовнішній ключ) |
| 3 | Text | текстовий | текст питання |
| 4 | A_ | текстовий | відповідь A |
| 5 | B_ | текстовий | відповідь B |
| 6 | C_ | текстовий | відповідь З |
| 7 | D_ | текстовий | відповідь D |
| 8 | Ans | безліч | Правильна відповідь (a, b, c, або d) |
| 9 | Typ_ctrl | безліч | Тип контролю (1 поточний, 2 поточний, ..., підсумковий) |
| 10 | Dat | Дата час | Дата останнього редагування |

Як бачимо кожне питання відноситься до того чи іншого типу контролю, таким

чином є можливість розділити питання по контролям - більш легкі питання віднести до поточних контролів, а складні до підсумкового або проміжного контролю.

Кожне питання має по чотири варіанти відповідей, а також поле «Ans», де вказано буквений ідентифікатор правильної відповіді.

**Таблиця «Студенти»**. Дана таблиця буде зберігати інформацію про зареєстровані в системі студентах. У таблиці 2.7 наведено структуру цієї таблиці:

Таблиця 2.7 – студенти

| **№** | **Назва поля** | **Тип поля** | **опис** |
|---|---|---|---|
| 1 | stud_ID | Лічильник | Числовий ідентифікатор студента (первинний ключ) |
| 2 | Fio | текстовий | П.І.Б. студента |
| 3 | Gr | Дата | Дата народження студента |
| 4 | Mail | текстовий | Адреса електронної пошти студента |
| 5 | Login | текстовий | Логін студента |
| 6 | Pass | текстовий | пароль студента |
| 7 | Dat | Дата час | Дата останнього редагування |

Видно, що ця таблиця за своєю структурою майже ідентична таблиці «Користувачі», за винятком того, що ці таблиці несуть в собі зовсім різні завдання.

**Таблиця «Результати»**. У цій таблиці буде зберігатися інформація про результати здачі контролів студентами з тих чи інших курсів. По суті, таблиця «Результати» буде електронної залікової книжкою студентів.

У таблиці 2.8 наведено структуру цієї таблиці:

Таблиця 2.8 – результати

| № | Назва поля | Тип поля | опис |
|---|---|---|---|
| 1 | Kurs_ID | Лічильник | Числовий ідентифікатор курсу (зовнішній ключ) |
| 2 | stud_ID | Лічильник | Числовий ідентифікатор студента (зовнішній ключ) |
| 3 | Tek1_bal | числовий | Набраний бал за 1 поточний контроль |
| 4 | Tek1_cnt | числовий | Кількість перездач 1 поточного контролю |
| 5 | Tek1_dat | Дата час | Дата / час останньої здачі (перездачі) 1 поточного контролю |
| 6 | Tek2_bal | числовий | Набраний бал за 2 поточний контроль |
| 7 | Tek2_cnt | числовий | Кількість перездач 2 поточного контролю |
| 8 | Tek2_dat | Дата час | Дата / час останньої здачі (перездачі) 2 поточного контролю |
| 9 | Tek3_bal | числовий | Набраний бал за 3 поточний контроль |
| 10 | Tek3_cnt | числовий | Кількість перездач 3 поточного контролю |
| 11 | Tek3_dat | Дата час | Дата / час останньої здачі (перездачі) 3 поточного контролю |
| 12 | Tek4_bal | числовий | Набраний бал за 4 поточний контроль |
| 13 | Tek4_cnt | числовий | Кількість перездач 4 поточного контролю |
| 14 | Tek4_dat | Дата час | Дата / час останньої здачі (перездачі) 4 поточного контролю |
| 15 | Prm1_bal | числовий | Набраний бал за 1 проміжний контроль |
| 16 | Prm1_cnt | числовий | Кількість перездач 1 проміжного контролю |
| 17 | Prm1_dat | Дата час | Дата / час останньої здачі (перездачі) 1 проміжного контролю |
| 18 | Prm2_bal | числовий | Набраний бал за 2 проміжний контроль |
| 19 | Prm2_cnt | числовий | Кількість перездач 2 проміжного контролю |
| 20 | Prm2_dat | Дата час | Дата / час останньої здачі (перездачі) 2 проміжного контролю |
| 21 | Itg_bal | числовий | Набраний бал за підсумковий контроль |
| 22 | Itg_cnt | числовий | Кількість перездач підсумкового контролю |
| 23 | Itg_dat | Дата час | Дата / час останньої здачі (перездачі) підсумкового контролю |

Всі поля даної таблиці досить інтуїтивно зрозумілі. Слід зазначити тільки те, що поля зберігають бали мають дійсний тип.

Програмне ядро тестового порталу в свою чергу ділиться на 3 блоки:

- Адміністраторський
- Операторський
- Студентський

Програмне ядро складається з наступних папок, які знаходяться в головному каталозі TEST_PORTAL, розташованому в кореневому веб-каталозі Apache-сервера (htdocs):

1)SQL. Містить єдиний файл - testportal_create.sql - який потрібен для створення первинної (порожній) бази даних порталу

2)ADM. Містить в собі скріптові файли необхідні для роботи адміністратора

3)OPER. Містить в собі скріптові файли необхідні для роботи операторів

4)IMAGES. Містить в собі графічні файли, необхідні для візуального інтерфейсу порталу

5)Студентська частина знаходиться в самому каталозі TEST_PORTAL

Розглянемо кожну частину окремо докладніше.

### 2.9 Адміністраторський блок

Найбільш об'ємна і важлива частина порталу з точки зору управління інформацією. До цієї частини має доступ тільки той, хто знає пароль адміністратора. Цей каталог містить наступні скриптові файли(таблиця 2.9):

Таблиця 2.9 – Файли адміністратора

| **Назва файлу** | **Короткий опис** |
|---|---|
| admin.php | Головний файл, що забезпечує інтерфейс адміністратора |
| adm_ch_data.php | Зміна персональних даних адміністратора |
| adm_ch_pass.php | Зміна пароля адміністратора |
| adm_logout.php | Забезпечення виходу адміністратора з системи |
| ch_acces.php | Редагування доступу до курсу |
| ch_acces_by_us.php | Редагування доступу до курсів щодо користувача |
| ch_ctrl_count.php | Редагування лімітів на кількість питань контролів |
| ch_kurs.php | Редагування інформації про курси |
| ch_limit.php | Редагування лімітів на перездачу |

| | |
|---|---|
| ch_quest.php | редагування питань |
| ch_user.php | Редагування власних даних |
| del_kurs.php | видалення курсів |
| del_quest.php | видалення питань |
| del_stud.php | видалення студентів |
| del_user.php | видалення користувачів |
| new_kurs.php | Додавання курсів |
| new_quest.php | Додавання питань |
| new_user.php | Додавання користувачів |
| output_fns.php | Файл, що відповідає за візуальне оформлення адмін-інтерфейсу |
| search_KURS.php | Пошук курсів |
| search_quest.php | Пошук питань |
| search_stud.php | Пошук студентів |
| search_user.php | Пошук користувачів |
| testportal_fns.php | Інклюд-файл, що включає інші необхідні файли |
| user_fns.php | Функції для роботи з БД |

Вся робота починається з файлу ADMIN.PHP. Це по суті індекс-файл адміністраторського блоку. Даний файл за умовчанням виводить на екран вікно авторизації адміністратора. Після введення логіна / пароля він перевіряє їх на коректність і після успішної авторизації виводить відповідний інтерфейс, в іншому випадку знову виводиться вікно авторизації з попереджуючим повідомленням про невірне паролі. Вся перевірка знаходиться в наступних рядках коду цього файлу:

```
if (isset ($ _ POST [ 'user']) && isset ($ _ POST [ 'password']))
{
$ User = $ _POST [ 'user'];
$ Pass = $ _POST [ 'password']; ($ user && $ pass)
{
{_Adm ($ user, $ pass);
$ _SESSION [ 'valid_adm'] = $ user;
$ Title = 'Тест-ПОРТАЛ :: Ласкаво просимо в адмін-модуль!';
$ Res = get_adm_data ();
$ Msg = '<font face = "Verdana" size = "2"> Ласкаво просимо в систему </ font>
```

*<font face = "Verdana" size = "2" color = "# FFFFFF">'*

*. $ Res [ 'fio']. "! </ Font>";*

*} (Exception $ e)*

*{_Main ( 'Тест-ПОРТАЛ :: Авторизація', 'Ви не ввійшли в систему! Авторизуйтесь!', $ Тур) ;;*

*}*

*}*

Скрипт перевіряє стан POST-змінних "*user*" і "*password*", і якщо вони встановлені, то передає їх функції log_adm, в якій і відбувається перевірка коректності даних.

Функція log_adm за отриманими даними повертає або True (вдала авторизація), або генерує виняткову помилку, яку в свою чергу відловлює вищевикладений оператор TRY. Якщо оператор TRY зловив помилку, то він відразу ж переходить до блоку catch, і заново виводить форму авторизації, в іншому випадку встановлюється сесійний змінна VALID_ADM, яка буде зберігати в собі протягом усієї сесії логін адміністратора. Ця сесійний змінна потрібна для того, щоб адміністратор (користувач) не вводив би кожен раз логін / пароль при переході на ту чи іншу посилання. Досить встановити таку змінну один раз при вдалій авторизації і вона буде передаватися в інші скрипти на перевірку, таким чином користувач буде позбавлений від нескінченних авторизаций, а також буде знижено навантаження на СУБД. Даний підхід практикується в багатьох сучасних Інтернет-додатках і є відносно швидким і безпечним. Після установки цієї змінної користувачеві відкривається вікно адміністраторського інтерфейсу, де доступні наступні функції: перегляд і зміна персональних даних та пароля адміністратора; пошук, додавання, видалення і редагування курсів, користувачів, питань, студентів.

Дуже важливою функцією, що вимагає забезпечення особливої безпеки є зміна пароля адміністратора. Цю функцію забезпечує файл adm_ch_pass.php. Даний файл

приймає три пост-змінні: поточний пароль, новий пароль і підтвердження нового пароля. Далі перевіряється не порожні ці змінні, тобто заповнив чи все поля користувач, і якщо хоча б одне з них не заповнено, то видається повідомлення про недостатність даних. Також перевіряється відповідність нового пароля і його підтвердження, якщо вони не ідентичні, то також видається відповідне повідомлення про помилку. Якщо ж всі умови задоволені, то змінні передаються функції adm_ch_pass (файл user_fns.php) і відбувається спроба зміни пароля адміністратора. Причому треба відзначити,

*«Update users set pass = sha1 ( '$ new_pass') where status = 'A' and pass = sha1 ( '$ old_pass') and login = '$ user'»*

Тобто зміна пароля відбувається у того користувача, який має статус «А» (адміністратор), пароль ідентичний поточному введеному паролю і логін рівний переданому, в нашому випадку логіну, який зберігається в сесійному змінної VALID_ADM.

Тут же народжується питання: «А що буде якщо з абсолютно іншої веб-сторінки передати ці три змінні, що пароль адміністратора зміниться?» Зрозуміло що ні, так як в цьому файлі перш, ніж взагалі проводити будь-які дії спочатку перевіряється увійшов адміністратор в систему, якщо це не так то скрипт нічого не буде робити і відобразить просто білий екран. Це завдання покладено на функцію check_adm (user_fns.php), яка включена в усі скріптові файли адміністраторського блоку:

```
function check_adm ()
{$ Valid_adm; (isset ($ _ SESSION [ 'valid_adm']))
{True;
}
}
```

Як бачимо, дана функція просто перевіряє чи встановлена сесійний змінна VALID_ADM, і в разі успіху повертає TRUE. Доступ до сесії відбувається за

допомогою оператора session_start (), який включений в самому початку кожного скрипта. Якщо забути написати даний оператор, то доступ до поточної сесії не буде здійснений і відповідно скрипт не працюватиме навіть при правильно введених даних.

Як було сказано вище - реалізація візуальної частини адміністраторського блоку покладено на скрипт output_fns.php. По суті даний файл на 70% складається з простого HTML-коду, але саме решта 30% PHP-коду відповідають за те, що і як буде відображатися. Хотілося б звернути увагу на найбільш цікаві і складні частини даного скрипта. Основний і часто використовуваний інструмент відображення - це посторінковий вивід інформації з бази даних. Так як обсяг виведеної інформації може досягати досить-таки значних розмірів, то чи не доцільно було б відображати відразу всю запитувану інформацію, набагато ефективніше, зручніше і швидше відображати інформацію посторінково - інформація ділиться на сторінки по n-записів на кожній, на кожній сторінці відображається чергова порція даних. Даний підхід може здатися на перший погляд складним і незрозумілим, але при правильно обраному алгоритмі реалізувати його досить просто. Розглянемо наприклад, функцію що виводить питання того чи іншого курсу get_quest (output_fns.php), інші функції забезпечують посторінковий вивід практично ідентичні цієї функції.

```
function get_quest ()
{
$ Num_q = 15;
$ Id = $ _GET [ 'id'];
$ Page = $ _GET [ 'page'];
$ Name = get_kurs_info ($ id) -> fetch_assoc ();
$ Name = $ name [ 'name'];
$ Res_q = get_kurs_quest ($ id, '0');
$ Cnt_q = $ res_q-> num_rows;
$ Total = intval (($ cnt_q - 1) / $ num_q) + 1;
```

*$ Page = intval ($ page);*

*if (empty ($ page) || $ page <0) $ page = 1; ($ Page> $ total) $ page = $ total;*

*$ Start = $ page * $ num_q - $ num_q;*

*$ Result = get_kurs_quest_by_page ($ id, $ start, $ num_q); ($ result-> num_rows> 0)*

*{ '<B> Поточний курс: </ b> <font face = "Verdana" size = "2">'. $ Name. '</ Font> <p>'; '<input type = "hidden" name = "page" value = " '. $ page.'"> ';' <table border = "1" width = "580" cellpadding = "0" cellspacing = "0"> <tr> <td height = "30" bgcolor = "# 333339"> <font face = "Verdana" size = "2" color = "white"> '.*

*'Питання курсу (Сторінок:'. $ Total. ':: Питань:'. $ Cnt_q. ':: Сторінка №:'. $ Page. ') </ Td> </ tr>';*

*for ($ i = 0; $ i <$ result-> num_rows; $ i ++)*

*{*

*$ Row = $ result-> fetch_assoc (); '<tr> <td> <table bgcolor = "white"> <tr> <td width = "580">'. Htmlspecialchars ($ row [ 'text']). '</ td> </ tr> </ td> </ tr>'; '<tr> <td align = "right"> <a target = "_ blank" href = "admin.php? typ = quest_info & id =' . $ row [ 'q_id'].*

*'& Page =". $ Page. ' "> <Font face =" Verdana "size =" 2 "> Докладно </a> & nbsp; & nbsp; <a href =" del_quest.php? Id =".*

*$ Row [ 'q_id']. ' "> <Font color =" red "face =" Verdana "size =" 2 "> Видалити </ td> </ tr> </ table>';*

*} '</ Table>';*

*if ($ page! = 1 && $ page-1! = 1) $ pervpage = '<a href=admin.php?typ=get_quest&id='.$id.'&page=1> Перша </a> |*

*<A href = admin.php? Typ = get_quest & id = '. $ Id.' & Page = '. ($ Page - 1). '> Попередня </a> |'; ($ Page! = $ Total && $ page + 1! = $ Total) $ nextpage = '| <A href = admin.php? Typ = get_quest & id = '. $ Id.' & Page = '. ($ Page + 1). '> Наступна </a> |*

*<A href = admin.php? Typ = get_quest & id = '. $ Id.' & Page = '. $ Total. '> Остання </a>'; ($ Page - 2> 0) $ page2left = '<a href = admin.php? Typ = get_quest & id =". $ Id. '& Page =". ($ Page - 2). '>'. ($ Page - 2). '</a> | ';*

*if ($ page - 1> 0) $ page1left = '<a href = admin.php? typ = get_quest & id =". $ id. '& page =". ($ Page - 1). '>'. ($ Page - 1). '</a> | '; ($ Page + 2 <= $ total) $ page2right = '| <A href = admin.php? Typ = get_quest & id = '. $ Id.' & Page = '. ($ Page + 2). '>'. ($ Page + 2). '</a>'; ($ Page + 1 <= $ total) $ page1right = '| <A href = admin.php? Typ = get_quest & id = '. $ Id.' & Page = '. ($ Page + 1). '>'. ($ Page + 1). '</a>'; $ Pervpage. $ Page2left. $ Page1left. '<B>'. $ Page. '</ B>'. $ Page1right. $ Page2right. $ Nextpage;*

*}*

*{ 'Питань за курсом не знайдено!';*

*}*

*}*

Розглянемо цю функцію через підрядник. Змінна $ num_q приймає кількість питань, яке потрібно відображати на кожній сторінці, в даному випадку - 15. В принципі це число можна змінювати на інше прийнятне. Далі приймається ідентифікатор курсу і сторінки, яку потрібно відобразити. Після чого згідно ідентифікатора курсу функція get_kurs_quest повертає кількість всіх питань курсу в змінну $ cnt_q. Використовуючи цю змінну за допомогою рядка

*$ Total = intval (($ cnt_q - 1) / $ num_q) + 1;*

ми вже дізнаємося скільки всього сторінок буде потрібно відобразити. Тут просто загальна кількість питань ділиться на кількість питань необхідне для відображення, причому функція intval дає можливість завжди отримувати тільки цілу частину отриманого числа, ми використовуємо її тут тому що в деяких випадках результат ($ cnt_q - 1) / $ num_q може виявитися речовим. Точно також ми вирізаємо з переданого ідентифікатора сторінки $ page речову частину, тому що скрипту може бути навмисне переданий речовинний ідентифікатор і без належної обробки висновок буде

викликати помилку. Далі ідентифікатор сторінки перевіряється на порожнє і негативне значення, якщо це сталося, то ідентифікатор сторінки автоматично скидається ну нуль. Точно також перевіряється чи не перевищує ідентифікатор максимальне число сторінок ($ total), якщо так, то ідентифікатор сторінки скидається на значення останньої сторінки. Потім вичсляем починаючи з якою записи слід виводити питання:

*$ Start = $ page * $ num_q - $ num_q;*

Функція get_kurs_quest_by_page (user_fns.php) повертає «порцію» питань відповідно до переданих їй параметрами (обмеженнями). Основна рядок цієї функції

*$ Result = $ conn-> query ( "select * from questions where kurs_id = '$ id' LIMIT $ start, $ num");*

повертає набір даних з таблиці «questions» (питання) складається з $ num записів, починаючи з записи $ start. Потім якщо кількість рядків більше нуля, тобто є чергова порція даних, то відбувається табличний висновок питань, в іншому випадку видається повідомлення про помилку. Ну і нарешті, буде потрібно висновок списку залишилися сторінок, якщо такі є. Наприклад, даний рядок перевіряє якщо сторінка не є першою і її значення зменшене на один також не є рівним одному, то значить потрібно вивести посилання на попередню і першу сторінки

*if ($ page! = 1 && $ page-1! = 1) $ pervpage = '<a href=admin.php?typ=get_quest&id='.$id.'&page=1> Перша </a> |*

*<A href = admin.php? Typ = get_quest & id = '. $ Id.' & Page = '. ($ Page - 1). '> Попередня </a> |';*

Майже також перевіряється і то, чи не потрібно вивести посилання на наступну і останню сторінки, а також виведення двох найближчих до поточної станиці сторінок. Таким чином нижнє меню навігації по сторінках може приймати приблизно такий вигляд:

*Перша| Попередня | 3 | 4 | 5 | Наступна | Остання*

Поточна сторінка відображається жирним шрифтом в нижньому меню, а також в заголовку таблиці. Як бачимо досить-таки не складний скрипт, який дозволяє швидко і зручно відображати великий обсяг інформації посторінково. Всі інші скрипти порталу для посторінкового виведення будь-якої інформації, практично ідентичні вищеописаного і відрізняються лише джерелом даних і візуальним оформленням.

Решта частини адміністраторського блоку по суті не представляють особливого інтересу з точки зору програмування, тому що в ньому немає особливих перевірок і обмежень в зв'язку з тим що цей блок орієнтований на одну людину має повний доступ до порталу. Хотілося б тільки відзначити пошук тієї чи іншої інформації порталу.

Як відомо пошук є невід'ємною частиною веб-систем мають справу з БД, якою є наш тестовий портал. Зокрема адміністратору надається можливість пошуку користувачів, питань, курсів і студентів за різними параметрами. Весь процес пошуку в основному лягає на SQL-запити функцій файлу user_fns.php. Розглянемо наприклад, пошук користувачів.

Пошук користувачів, так само як і інший інформації, можна зробити по одному або декільком параметрам. Пошук користувачів можна зробити по П.І.Б., адресу електронної пошти та логіну, або варіації вищезазначених параметрів. Введені дані передаються скрипту search_user.php, який в свою чергу перевіряє - чи є хоча б один переданий параметр і в разі успіху передає параметр (и) в функцію user_search. Ця функція за переданими їй параметрам повертає набір даних. Наприклад, якщо передані всі три параметри (П.І.Б., адреса електронної пошти, логін), то запит цієї функції виглядає наступним чином:

*$ Quer = "select * from users where fio like '% $ fio%' and mail like '% $ mail%' and login '% $ log%' and status <> 'A'";*

Даний запит включає в себе SQL-функцію LIKE, яка повертає дані з БД згідно

строкового параметра. Наприклад, фрагмент «fio like '% ІВАНо%'» означає, що запит поверне дані, в яких значення поля FIO містять в собі набір букв «ІВАН», причому неважливо - в кінці, середині або на початку даних. Якщо поставити знак «%» тільки спочатку параметра, то умова б звучало приблизно так: «повернути всі записи, в яких значення поля FiO мають в кінці набір букв 'ІВАН'». У нашому ж випадку в результуючий набір даних можуть потрапити дані види:

*Іван*

*Лівані*

*дивані*

*ІВАН*

Як бачимо, досить-таки зручна функція дозволяє проводити пошук на частковий збіг, що важливо при великих і схожих один на одного даних. Причому як можна помітити, функції не розбірлива до регістру букв, тобто при пошуку значення «Іван» буде рівносильно «ІВАН» і т.п. Зрозуміло пошук можна було б проводити на точний збіг, прибравши функцію LIKE, але де наприклад, гарантія того, що адміністратор (користувач) НЕ помилиться в одному-двох символах при введенні чергової інформації?

Як і пошук користувачів пошук будь-який інший інформації в порталі заснований саме на такому підході і відрізняється лише джерелом даних.

Нарешті, розглянемо як відбувається вихід із системи. За вихід адміністратора з системи відповідає скрипт adm_logout.php. Даний файл підключається до поточної сесії і витягує з неї сесійний змінну valid_adm, після чого стирає її з сесії:

*unset ($ _ SESSION [ 'valid_adm']);*

а потім повністю «вбиває» всю сесію:

*$ Resul_dest = session_destroy ();*

Після чого на екрані знову з'являється форма авторизації і доступ до всіх файлів недоступний до тих пір поки адміністратор (користувач) знов не авторизується.

Аналогічним чином відбувається вихід із системи в операторському і студентському блоках.

Отже в даному підрозділі були розглянуті основні моменти програмної частини адміністраторського блоку, перейдемо тепер до розгляду операторського блоку.

### 2.10 Операторський блок

Та частина порталу, яка відповідає за ведення БД порталу. Саме оператори - є основний персонал, який повинен працювати з системами подібного роду, тому що погодьтеся не буде ж один адміністратор вводити курси, питання, виправляти помилки і т.д. Для цього і створений операторський блок. Сюди має доступ, тільки той персонал, якому адміністратор відкрив обліковий запис оператора і дав доступ до тих чи інших ділянках порталу. Цей каталог містить наступні скріптові файли:

Таблиця 2.10 – Операторський блок

| **Назва файлу** | **Короткий опис** |
|---|---|
| oper.php | Головний файл, що забезпечує інтерфейс оператора |
| ch_data.php | Зміна персональних даних оператора |
| ch_pass.php | Зміна пароля оператора |
| logout.php | Забезпечення виходу оператора з системи |
| ch_limit.php | Редагування лімітів на перездачу |
| ch_quest.php | редагування питань |
| del_quest.php | видалення питань |
| new_quest.php | Додавання питань |
| Output.php | Файл, що відповідає за візуальне оформлення оператор-інтерфейсу |
| Oper_fns.php | Функції для роботи з БД |
| Oper_inc.php | Інклюд-файл, що включає інші необхідні файли |

Індекс-файлом операторського блоку є файл - oper.php. Інтерфейс оператора будується на основі його доступу, тобто оператору будуть доступні тільки ті функції доступ до яких йому дозволено. Первісне меню однаково для всіх операторів і складається з наступних пунктів: персональні дані, зміна пароля, доступ і вихід. Все меню практично ідентичні з однойменними меню адміністратора за винятком меню

«Доступ» - саме через нього оператор веде всю роботу.

Після авторизації оператора як і в випадку з адміністратором встановлюється сесійний змінна valid_oper містить в собі логін авторизованого оператора, але крім логіна встановлюється змінна valid_oper_id, яка зберігає протягом сесії ідентифікатор авторизованого оператора. Ідентифікатор оператора потрібен для перевірок доступу оператора, тому що все що пов'язано з доступом реалізується через ідентифікатори операторів з принципів мінімізації БД.

Отже, меню «Доступ». Це меню посторінково показує список тих курсів, до яких оператор має хоча б мінімальний доступ (доступ на читання). Це завдання реалізується функцією show_enabled_kurses (output.php). Головною рядки тут є рядок

*$ Result = get_kurs_by_page ($ oper_id, $ start, $ num_k, true);*

У змінну $ result повертається список доступних оператору курсів за допомогою функції get_kurs_by_page. Цій функції передається ідентифікатор оператора, потім проводиться запит наступного виду:

*"Select * from kurs, dostup where dostup.kurs_id = kurs.kurs_ID and dostup.us_ID = $ us_id and sel = '1'"*

Даний запит за ідентифікатором оператора звіряє його доступ до всіх курсів і повертає тільки ті курси, до яких оператор має доступ хоча б на читання (sel = '1'). Далі функція show_enabled_kurses перевіряє, чи є хоча б один рядок в результуючому наборі даних (є доступні курси):

*if ($ result-> num_rows> 0)*

якщо це так, то відображається таблиця доступних курсів, в іншому випадку видається повідомлення про те, що оператор не має доступ до жодного з курсів.

При натисканні вже на конкретний курс буде відображений інтерфейс роботи з даними курсом. Знову таки тут все залежить від ступеня доступу оператора. За замовчуванням у оператора при роботі з конкретним курсом є мінімальний набір інформації: інформація про курс (найменування, тьютор), кількість питань курсу,

інформація про ліміти курсу і список питань курсу. Тут все зрозуміло, набагато цікавіше розглянути ситуацію коли оператор має доступ більш ніж тільки на читання. Почнемо з лімітів курсу. При виведенні лімітів курсу, відбувається перевірка доступу оператора до лімітів - функція show_kurs_access (output.php). Спочатку ми вдруге перевіряємо доступ оператора до курсу:

*$ Oper = $ _SESSION [ 'valid_oper_id'];*

*$ Res = get_kurs ($ id, $ oper);*

тому на момент переходу за посиланням може статися так, що адміністрація відключила можливість використовувати даному оператору доступ. Потім якщо доступ як і раніше є, то в змінну $ res_2 зберігається докладна інформація про доступ оператора до даного курсу:

*$ Res = $ res-> fetch_assoc ();*

*$ Res_2 = $ res;*

Далі перевіряємо конкретно має оператор доступ до лімітів курсу на перездачу, і якщо це так то виводимо тег форми, яка взаємодіє зі скриптом ch_limit.php.

*if ($ res_2 [ 'lim'])*

*echo '<form method = "post" action = "ch_limit.php"> <input type = "hidden" name = "id" value ='. $ id. '>';*

Після закінчення виведення інформації про ліміти, в разі дозволеного доступу виводиться кнопка «Застосувати»:

*if ($ res_2 [ 'lim']) echo '<tr> <td align = "right"> <input type = "submit" value = "Застосувати"> & nbsp </ td> </ tr>';*

За допомогою даної кнопки оператор матиме можливість змінювати ліміти на перездачу. При натисканні кнопки введені дані передаються скрипту ch_limit.php. Якщо ж оператор не має даного доступу, то кнопка «Застосувати» не з'явиться, і оператор всього лише може переглядати ліміти.

Наступним важливим моментом є висновок меню для додавання питань. Це такі

рядки все тієї ж функції show_kurs_access (output.php):

*if ($ res_2 [ 'ins']) '<table width = "600"> <tr> <td> <a target = "_ blank" title = "Додавання питання по обраному курсу" href = oper.php? typ = new_quest & id = '. $ id.'> Додати питання </a> </ td> </ tr> </ table> ';*

все просто - перевіряється чи має оператор доступ на вставку питань ($ res_2 [ 'ins']) то виводиться посилання «Додати питання».

Ну і нарешті, список питань. При виведенні списку за замовчуванням є меню «Докладно», але також може бути доступним меню «Видалити», зрозуміло якщо оператор має доступ на видалення. Висновок питань відбувається в тій же функції, що і висновок лімітів. Тут головною рядком перевірки є рядок:

*if ($ res_2 [ 'del']) $ del = '& nbsp; & nbsp; <a href="del_quest.php?id='.$row['q_id'].'"> <font color = "red" face = "Verdana" size = "2"> Видалити '; $ del =' ';*

Якщо оператор має доступ на видалення ($ res_2 [ 'del']), то в змінну $ del заноситься тег виведення посилання для видалення питання. Це посилання передає скрипту del_quest.php параметр id (ідентифікатор питання), який потрібно видалити. Далі при виведенні меню питання до стандартного пункту «Докладно» просто приєднується строкова змінна $ del (або вона порожня, або вона містить посилання на видалення):

*echo '<tr> <td align = "right"> <a target="_blank" href="oper.php?typ=quest_info&id='.$row['q_id'].'"> <font face = "Verdana "size =" 2 "> Докладно </a> '. $ del.' </ td> </ tr> </ table> ';*

Посилання «Докладно» тут дає можливість переглянути інформацію про конкретному питанні в новому вікні. Причому в цьому новому вікні знову ж залежно від доступу оператора може з'явитися кнопка «Застосувати» дл редагування питання. Код перевірки доступу на редагування практично ідентичний з кодом перевірки доступу до лімітів курсу, тому розглядати його немає особливої потреби.

Окремої уваги при роботі оператора займає перевірка даних, що вводяться і коректне відображення в браузері. Відомо, що злом будь веб-системи часто відбувається через те, що розробники допускають на перший погляд дрібні недоліки при маніпулюванні з даними. Розглянемо можливості злому на прикладі нашого порталу. Припустимо при введенні питання не відбувається ніякої обробки текстових даних питання, тоді зловмисник просто напросто введе в поле тексту питання невеликий скрипт для злому. Далі цей скрипт буде збережений в БД як звичайний питання, але при відображенні в браузері цей нібито «звичайний» питання буде вже не просто відображатися, а виробляти певні дії, не передбачені розробниками. Не важко уявити, що буде якщо таке допустити. Для таких випадків у мові PHP був введений ряд спеціальних операторів, обмежують ті чи інші дії. Ці оператори зустрічаються в лістингах тестового порталу практично скрізь, де відбуваються маніпуляції з текстовими даними.

Перший і дуже часто зустрічається оператор - це оператор «Addslashes». Потрібен він для того, щоб скасовувати деякі керуючі структури, до таких належать лапки (одинарні та подвійні), символ зворотної косої риски (\) і символ NULL. Ці символи цілком можуть бути частиною тексту, але при занесенні такого тексту в БД, можуть виникнути проблеми неправильної інтерпретації цих символів з боку СУБД "MySQL". Тому до збереження подібного роду текстових даних в БД слід застосовувати цей оператор. Зрозуміло дані будуть збережені в дещо іншому вигляді, тому для правильного відображення таких даних в браузері треба перетворити їх до первісного вигляду за допомогою оператора stripslashes - який є повним протидією оператору addslashes. Наступний дуже корисний оператор, який часто буває потрібен для фільтрування даних це оператор - striptags (). Цей оператор повністю видаляє з переданої йому рядки все HTML- і PHP-дескриптори, що не дозволить створювати сценарії всередині даних, які сервер передає для відображення браузеру. Припустимо змінна $ str містить наступний текст:

«Сценарій хакера; require_once ( 'www.hack.ru/exmlp1.php'); echo list (this); »
Те після застосування до неї оператора striptags (), вона прийме такий вигляд%
«Сценарій хакера;»

Як бачимо дуже корисна і потрібна функція. Її можна використовувати для обробки тих, даних які свідомо не повинні містити веб-дескриптори - П.І.Б., адреса, логіни і т.п. Зрозуміло недоцільно використовувати її, наприклад для обробки тексту питань тестового порталу, тому що цілком можливо що питання може містити в собі фрагменти, що мають HTML- і PHP-дескриптори. Як бути в такому випадку? По-перше, при збереженні такого роду тексту знову ж слід застосувати вищеописані оператори скасування керуючих структур, що і робиться в функції add_quest (oper_fns.php), яка додає новий питання:

*$ Text = addslashes (substr (trim ($ text), 0,255));*

*$ A = addslashes (substr (trim ($ a), 0,255));*

*$ B = addslashes (substr (trim ($ b), 0,255));*

*$ C = addslashes (substr (trim ($ c), 0,255));*

*$ D = addslashes (substr (trim ($ d), 0,255));*

Звичайно ж цими рядками проблема не вирішена, а вирішується вона при відображенні даних у браузері за допомогою оператора htmlspecialchars (), який призначений для перетворення веб-дескрипторів в безпечні текстові еквіваленти. Даний підхід можна побачити наприклад, у функції quest_info (output.php), яка відображає в браузері дані питання, в тому числі - його текст і відповіді:

*<Td> Текст питання: </ td> <td> <textarea rows = "5" name = "text" cols = "55" maxlength = "10"> <? Php echo htmlspecialchars (trim (stripslashes ($ res [ ' text ']))))?> </ textarea> </ tr> <tr> <td> Відповідь A: </ td> <td> <textarea rows = "2" name = "a" cols = "55" maxlength = "10"> <? php echo htmlspecialchars (trim (stripslashes ($ res [ 'a _']))))?> </ textarea> </ td> <tr> <td> Відповідь B: </ td> <td > <textarea rows = "2" name = "b" cols = "55" maxlength = "10"> <? php echo htmlspecialchars (trim*

*(stripslashes ($ res [ 'b _']))?> </ textarea> </ td> <tr> <td> Відповідь C: </ td> <td> <textarea rows = "2" name = "c"cols = "55" maxlength = "10"> <? php echo htmlspecialchars (trim (stripslashes ($ res [ 'c _']))?> </ textarea> </ td> <tr> <td> Відповідь D: </ td> <td> <textarea rows = "2" name = "d" cols = "55" maxlength = "10"> <? php echo htmlspecialchars (trim (stripslashes ($ res [ 'd_']))) ?> </ textarea> </ td>*

Таким чином забезпечується безпека при введенні і відображенні даних тестового порталу. Хоча звичайно це далеко не межа, і перевірки можна ускладнити і посилити. Але і цього цілком достатньо для того, щоб злом і псування даних була зведена до мінімуму.

### 2.11 Студентський блок

Даний блок є тим ядром, в якому відбувається весь процес взаємодії студентів з порталом, тут же відбувається реєстрація нових студентів. Цей каталог містить наступні скріптові файли:

Таблиця 2.11 – Студентський блок

| **Назва файлу** | **Короткий опис** |
|---|---|
| index.php | Головний файл, що забезпечує інтерфейс студентів |
| ch_data.php | Зміна персональних даних студента |
| ch_pass.php | Зміна пароля студента |
| logout.php | Забезпечення виходу студента з системи |
| registr.php | Реєстрація нового студента |
| check_result.php | Перевірка результатів пройденого контролю |
| db_fns.php | Файл підключення до БД (використовується всіма блоками програмного ядра) |
| Output.php | Файл, що відповідає за візуальне оформлення студент-інтерфейсу |
| student_fns.php | Функції для роботи з БД |
| student_inc.php | Інклюд-файл, що включає інші необхідні файли |

Головним файлом зрозуміло є index.php. За замовчуванням виводиться форма авторизації зареєстрованих студентів, що вимагає введення логіна і пароля. Якщо студент ще не зареєстрований, то потрібно перейти за посиланням «Реєстрація».

Після переходу на сторінку реєстрації і введення нових даних, ці дані передаються скрипту registr.php на перевірку. Даний файл фільтрує введені дані

*$ Fio = addslashes (strip_tags ($ _ POST [ 'fio']));*

*$ Mail = addslashes (strip_tags ($ _ POST [ 'mail']));*

*$ Log = addslashes (strip_tags ($ _ POST [ 'log']));*

Як бачимо з полів «ПІБ», «E-mail» і «Логін» видаляються керуючі символи, а також веб-дескриптори. Потім перевіряється валідність числа:

*$ Ch = (int) $ ch;*

*$ God = (int) $ god;*

*$ Mes = (int) $ mes; (! Checkdate ($ mes, $ ch, $ god))*

*{_Main ($ title, $ url. ':: Реєстрація неможлива! Неіснуюча дата! ::'. $ Url, 'reg') ;;*

*}*

Поля «Число», «Місяць» та «Рік» конвертуються в цілі числа і за допомогою php-функції checkdate перевіряється чи реально існує введена дата, якщо це не так то видається повідомлення про помилку. Потім за допомогою функції check_new_stud (student_fns.php) перевіряється чи не зайнятий вже логін, який був тільки що введений:

*function check_new_stud ($ user)*

*{*

*$ Conn = db_connect ();*

*$ Conn-> query ( "SET NAMES 'cp1251'");*

*$ Result = $ conn-> query ( "select * from students where login = '$ user'"); (! Result) new Exception ( 'помилка! Не можу перевірити дані, що вводяться!');*

*if ($ result-> num_rows> 0) false;*

*elsetrue;*

*}*

Ця функція запитує з таблиці студентів запис має такий же логін, що і переданий функції як параметр, якщо кількість повернутих рядків більше нуля функція повертає FALSE, тим самим сповіщаючи програму про те, що даний логін зайнятий. Якщо ж логін чи не зайнятий, то введені дані зберігаються в БД і пропонується вхід в систему під введеним логіном і паролем.

Після входячи в систему з'являється інтерфейс студентського блоку, де найбільш важливим пунктом меню є меню «Курси». Дане меню відображає всі активні курси, тобто ті курси за якими в даний момент можна пройти тестові контролі. Після вибору конкретного курсу, відображається меню для проходження контролів даного курсу. Відображення такого меню забезпечує функція show_kurs_ctrl (output.php). На початку дана функція отримує дані про ліміти курсу, для подальшого їх відображення в таблиці:

```
$ Kurs_id = $ _GET [ 'id'];
$ Stud_id = $ _SESSION [ 'valid_stud_id'];
$ Res = get_stud_ctrls ($ stud_id, $ kurs_id, true);
$ Lim_res = get_kurs_lim_for_stud ($ kurs_id);
$ Itog = 0; ($ lim_res-> num_rows> 0)
{
$ Lim_res = $ lim_res-> fetch_assoc ();
$ Lim [1] = $ lim_res [ 'tek1']; // ліміти на перездачу 1-го поточного контролю
$ Lim [2] = $ lim_res [ 'tek2']; //і т.д.
$ Lim [3] = $ lim_res [ 'tek3'];
$ Lim [4] = $ lim_res [ 'tek4'];
$ Lim [5] = $ lim_res [ 'prm1'];
$ Lim [6] = $ lim_res [ 'prm2'];
$ Lim [7] = $ lim_res [ 'itg'];
}
```

Після чого виводиться таблиця про пройдені і не пройдених контролях з даного курсу. Змінна $ res містить в собі набір даних щодо результатів студента по курсу, якщо кількість її рядків більше нуля (значить студент вже проходив деякі контролі даного курсу), таблиця виводиться відповідно до інформації про результати студента. У цьому випадку основне навантаження при виведенні таблиці несуть рядки типу (функція show_kurs_ctrl):

*if (! check_stud_ctrls ($ stud_id, $ kurs_id, 'tek2')) '2-й поточний'; echo '<a href="index.php?typ=go_ctrl&ctrl=tek2&id='.$kurs_id.'"> 2-й поточний </a>';*

Що робить функція check_stud_ctrls? Ця функція отримує ідентифікатор студента, курсу і тип контролю, а потім перевіряє чи має право студент проходити переданий тип контролю, і якщо так то виводить відповідне посилання, в іншому випадку телефон просто найменування чергового контролю. Таким чином студент не зможе проходити черговий контроль, не пройшовши попередній, що є певним стимулом для більш глибокого вивчення студентом матеріалу по курсу.

Якщо ж змінна $ res не містить жодного рядка, то це означає, що студент не проходив жодної контролю за цим курсом і виводиться стандартна таблиця з єдиною посиланням на перший поточний контроль.

Таким чином, якщо студент має право пройти черговий контроль за цим курсом, то цей контроль буде відображений у вигляді посилання. Навіть якщо студент спробує обдурити скрипт і свідомо відіслати йому той тип контролю, який на даний момент він не має право здати, то він отримає повідомлення про заборону, тому що перш ніж завантажити питання даного контролю функція show_ctrl (output.fns) перевіряє чи має студент право на проходження даного контролю:

*$ Id = $ _GET [ 'id'];*

*$ Ctrl = $ _GET [ 'ctrl'];*

*$ Stud_id = $ _SESSION [ 'valid_stud_id'];*

*$ Kurs = get_kurs ($ id) -> fetch_assoc ();*

*$ Kurs = $ kurs [ 'name']; '<b> Курс: </ b>'. $ Kurs. '<P>'; (! Check_stud_ctrls ($ stud_id, $ id, $ ctrl))*

*{ 'У Вас немає прав пройти даний контроль! <P> <b> Причина: </ b> Чи не пройдені попередні контролі!'; _ Str () ;;*

*}*

Далі в цій же функції ми бачимо наступну ділянку коду:

*if (! check_ctrl_quest_cnt ($ id, $ ctrl))*

*{ 'На жаль в даний момент неможливо пройти контроль! <P> <b> Можливі причини: </ b> <p> 1) Брак питань <p> 2) Не встановлені ліміти';*

*}*

Даний фрагмент перевіряє - чи вистачає в базі даних питань за даним типом контролю цього курсу. Якщо питань у БД недостатньо, то видається відповідне повідомлення. Ця функція приймає два параметри: ідентифікатор курсу і тип контролю:

*function check_ctrl_quest_cnt ($ id, $ ctrl)*

*{(Check_stud ())*

*{*

*$ Conn = db_connect ();*

*$ Conn-> query ( "SET NAMES 'cp1251'");*

*$ Cnt = $ conn-> query ( "select count (q_id) as cnt from questions where kurs_id = '$ id' and typ_ctrl = '$ ctrl'") -> fetch_assoc (); ($ cnt [ 'cnt'] < 1 || empty ($ cnt))*

*{False ;;*

*}*

*$ Lim = $ conn-> query ( "select * from ctrl_counts where kurs_id = '$ id'"); ($ lim-> num_rows> 0)*

*{*

*$ Lim = $ lim-> fetch_assoc (); ($ cnt [ 'cnt']> $ lim [$ ctrl] * 2) true; false;*

*} False;*

*}*

*}*

Функція обчислює кількість питань за даним типом контролю згідно ідентифікатора курсу ($ cnt), потім отримує значення кількості питань, яке повинно виводитися з даного контролю цього курсу (таблиця «Ліміти на кількість питань»). Після чого перевіряється - якщо реальна кількість питань в БД більше, ніж значення ліміту помножене на 2, то повертається TRUE і студенту відображається форма для проходження тестів. Умова «if ($ cnt [ 'cnt']> $ lim [$ ctrl] * 2)» звичайно взято довільно, в реальності кількість питань повинна перевищувати ліміт у багато разів - для того щоб, при кожному проходженні даного контролю не було повторення тих чи інших питань. Тут все просто - чим більше питань, тим менше ступінь повторення, що важливо для систем подібного роду.

Продовжимо розгляд функції show_ctrl (output.fns). Трохи нижче останнього описаного фрагмента йде наступний фрагмент:

```
if (get_ctrl_lim ($ id))
{
$ Lim = get_ctrl_lim ($ id) -> fetch_assoc ();
$ Cnt_lim = $ lim [$ ctrl]; (get_stud_res ($ stud_id, $ id))
{
$ Stud_res_lim = get_stud_res ($ stud_id, $ id) -> fetch_assoc (); (($ stud_res_lim [$ ctrl .'_ cnt ']) == $ cnt_lim)
{ '<B> Неможливо перездати контроль, тому що вичерпано ліміт перездач на даний контроль! </ b> ';
down_str () ;;
} (($ Stud_res_lim [$ ctrl .'_ cnt '] + 1) == $ cnt_lim)' <font face = "Verdana" size = "2"> <b> УВАГА! Це Ваша остання спроба перездати даний контроль, тому будьте
```

уважні при відповідях! </ B> </ font> ';

} ($ Cnt_lim == 0) '<font face = "Verdana" size = "2"> <b> УВАГА! Даний контроль не дозволено перездавати, тому будьте уважні при відповідях! </ B> </ font> <p> ';

}

Тут функція get_ctrl_lim перевіряє встановлені ліміти на перездачу даного контролю. Якщо немає, то значить студенти можуть перездавати даний контроль необмежену кількість разів. В іншому випадку йде наступна перевірка:

за допомогою функції get_stud_res (student_fns.php) виходять дані про хід здачі контролів студентом з даного курсу, потім порівнюється кількість перездач студентом даного контролю з лімітом на перездачу даного контролю (if (($ stud_res_lim [$ ctrl .'_ cnt ']) == $ cnt_lim)) - якщо обидва кількості рівні, то це говорить про те, що студент досяг ліміту на перездачу і видається повідомлення про неможливість перездачі. Якщо ж ліміт на перездачу дорівнює нулю (if ($ cnt_lim == 0)), то видається відповідне попередження, щоб студенти були більш уважні при відповідях. На цьому все перевірки, що відбуваються перед виведенням питань закінчені. Перейдемо до розгляду скрипта виводить питання.

Висновок питань відбувається всередині все тієї ж функції show_ctrl (output.fns):

*if (get_ctrl_quest_cnt ($ id))*

*{*

*$ Cnt = get_ctrl_quest_cnt ($ id) -> fetch_assoc ();*

*$ Cnt = $ cnt [$ ctrl]; // кількість виведених питань згідно ліміту з питань*

*$ Quest = get_quest_by_lim ($ id, $ ctrl, $ cnt);*

Спочатку отримуємо ліміт на кількість виведених питань, тобто дізнаємося скільки питань потрібно вивести для даного контролю. Потім функція get_quest_by_lim повертає стільки питань скільки потрібно, тобто стільки скільки

визначено в ліміті:

*function get_quest_by_lim ($ id, $ ctrl, $ lim)*

*{*

*if (check_stud ())*

*{*

*$ Conn = db_connect ();*

*$ Conn-> query ( "SET NAMES 'cp1251'");*

*$ Result = $ conn-> query ( "select * from questions where kurs_id = '$ id' and typ_ctrl = '$ ctrl' order by RAND () LIMIT 0, $ lim"); ($ result-> num_rows> = $ lim) $ result; false;*

*}*

*}*

Тут знову ж використана конструкція LIMIT, причому слід звернути увагу на те, що запит також містить конструкцію «order by RAND ()», завдяки якій завжди повертається випадковий набір записів, зрозуміло ступінь випадковості знову-таки залежить від кількості питань! Таким чином маючи велику кількість питань, студент при проходженні одного і того ж контролю кілька разів буде мати справу з новими питаннями.

Ну і нарешті, найцікавіший скрипт - скрипт перевірки правильності відповідей check_result.php. Даний скрипт приймає ідентифікатор курсу, тип контролю, ідентифікатори питань і відповіді на них. Тип контролю потрібен для вторинної перевірки того - чи має студент право здавати даний контроль чи ні. Далі перевіряється активний чи курс, тому що на момент відправки відповідей, може виникнути ситуація коли курс деактивували, в цьому випадку відповіді не можуть зараховуватися і видається відповідне повідомлення:

*if (check_stud)*

*{(! Get_kurs ($ id))*

*{ 'Курс не найден!';*

*exit;*

*}*

*{*

*$ Row = get_kurs ($ id) -> fetch_assoc ();*

*if ($ row [ 'act']! = 1)*

*{_Main ($ title, 'На жаль в даний момент курс не активний! Ваші результати не можуть бути перевірені!', '') ;; }}*

Після цього скрипт вдруге перевіряє не вичерпаний ліміт на перездачу, якщо це так, то знову ж відповіді не зараховуються і видається повідомлення про заборону. Якщо ж все в порядку, то скрипт перевіряє - чи на всі питання відповів студент:

*foreach ($ quest as $ q)*

*{*

*$ Ans [$ i] = $ _POST [$ q]; (empty ($ ans [$ i]))*

*{*

*$ S = '<font face = "Verdana" size = "2"> <a href=index.php?typ=go_ctrl&ctrl='.$ctrl.'&id='.$id.'> << Повернутися </ a > ';*

*show_main ($ title, $ s. ':: Неможливо перевірити результат! Дайте відповідь на ВСІ питання!', '') ;;*

*}*

*$ I ++;*

*}*

Масив $ ans тут містить надіслані відповіді, а масив $ quest - ідентифікатори відображених питань. Якщо хоча б один член масиву відповідей порожній, то значить студент не відповів на всі питання і в результаті цього відповіді не можуть бути зараховані. В іншому випадку ведеться підрахунок правильних відповідей:

*foreach ($ quest as $ q)*

*{(Check_ans ($ q, $ ans [$ i]))*

*$ Cnt ++; // кількість правильних відповідей*

*$ I ++;*

*}*

*$ Result = intval ($ cnt / $ i * 100);*

Тут основна робота лягає на функцію check_ans (student_fns.php), яка за ідентифікатором питання і відповіді видає результат перевірки:

*function check_ans ($ q_id, $ ans)*

*{(Check_stud ())*

*{*

*$ Conn = db_connect ();*

*$ Conn-> query ( "SET NAMES 'cp1251'");*

*$ Result = $ conn-> query ( "select * from questions where q_id = '$ q_id' and ans = '$ ans'"); ($ result-> num_rows> 0) true;*

*elsefalse;*

*}*

*}*

Якщо в таблиці питань знайдений питання з переданим ідентифікатором і правильною відповіддю рівним параметру $ ans, то значить відповідь правильна і результат функції TRUE, в іншому випадку - повертається FALSE.

У підсумку результат здачі контролю обчислюється дуже простою формулою: обчислюємо процентне співвідношення загальної кількості питань з кількістю правильних відповідей і отримуємо результат у вигляді відсотків. Отриманий результат функція set_ctrl_res (student_fns.php) записує результат здачі в таблицю результатів, причому інкрементує кількість перездач на одиницю, якщо контроль здається не перший раз. Потім відображається результат здачі тестів у вигляді набраних балів і словесної оцінки (незадовільно, задовільно, добре, відмінно).Ось в

принципі, всі основні моменти студентського блоку. Чи не був розглянутий пункт меню «заліковку», тому що він всього-на-всього відображає результати студента за всіма контролям в розрізі курсів, зрозуміло велика частина цього скрипта вдає із себе HTML-дескриптори.

### 2.12 Модуль контролю студентів

Для попередження списування студентами відповідей, розробимо та встановимо до системи деякий модуль, який дасть змогу перевірити , чи не списував випробовуваний відповіді. Для цього використаємо веб камеру, метод getUserMedia, та сервер на який буде відправлено запис проведення тестування. Працює це наступним чином:

1) Студент починає тест
2) Система за допомогою методу “getUserMedia” вмикає веб камеру, та починає запис тестування
3) Студент завершує тест, натискає клавішу сдачі тесту, веб камера відключається, запис тестування відправляється на локальний сервер для зберігання.
4) Якщо у викладача виникли сумніви, щодо чесності та об’єктивності зданого студентом тесту, він в будь який момент часу може отримати доступ до запису тестування цього студента, переглянути його, та зробити висновок щодо коректності оцінки студента.

Варто зазначити, що веб камераи (навіть вимкнена),лише своєю наявністю дозволить відбити студентам бажання здати іспит нечесно. Тут можна провести паралель з камерами реєстрації порушень на дорогах, більшість водіїв відразу знижують швидкість, та починають вести автомобіль більш обережно, як тільки побачать дорожній знак, не знаючи працює камера в даний момент чи ні.

# 3 АПРОБАЦІЯ РОЗРОБЛЕНОЇ СИСТЕМИ

## 3.1 Огляд адмін-модулю

Для роботи з адміністратративним блоком необхідно ввести в адресний рядок браузера наступне: *“http: //localhost/test_portal/adm/admin.php”*. Після чого перед нами з'явиться вікно авторизації для входу в адмін-модуль (рисунок 3.1). За замовчуванням логін / пароль адміністратора мають значення admin / admin. Зрозуміло рекомендується відразу після установки порталу змінити пароль, логін «вшитий» в базу даних і не може бути змінено!

Рисунок 3.1 – Головна сторінка

Рисунок 3.2 – Меню адміністратора

Після успішної авторизації перед нами з'явиться інтерфейс адміністратора (рисунок 3.2). Тут в лівому кутку є основне меню для маніпулювання даними порталу. Меню «Персональні дані» дозволяє змінити прізвище, ім'я, по батькові та e-mail адміністратора за допомогою введення нових даних і натискання кнопки «Застосувати» (малюнок 3.3).

Рисунок 3.3 - Авторизація

Далі є меню для зміни пароля, після натискання на нього відкриється спеціальна форма для введення нового пароля (рисунок 3.4):

Рисунок 3.4 - Вікно зміни пароля адміністратора

При натисканні на посилання «Додати користувача» відкриється форма для введення нового оператора порталу. Тут потрібно ввести прізвище, ім'я, по батькові, e-mail, логін і пароль оператора (рисунок 3.5):

Рисунок 3.5 – Вікно добавлення нового оператора

Причому якщо в базі вже є оператор з тільки що введеним логіном, то новий оператор не буде додано і видасться відповідне повідомлення про дублікації логіна. Всіх користувачів порталу можна переглянути через меню «Користувачі» - з'явиться сторінка зі списком операторів порталу, подібна зображеної на малюнку 4.4.6:

Рисунок 3.6 - Сторінка «юзерів»

На цій сторінці так само є можливість видалення операторів як всіх відразу, так і вибірково за допомогою кнопки «Видалити» і відповідних позначок.

За допомогою меню «Пошук користувачів» адміністратор може знайти операторів порталу згідно введених параметрів запиту пошуку. Пошук можна зробити по одному або декільком параметрам, зокрема - по П.І.Б., адресу електронної пошти та логіну (рисунок 3.7). Після натискання на кнопку «Пошук», якщо в базі даних порталу є оператори з зазначеними даними, з'явиться сторінка зі списком знайдених користувачів. В іншому випадку з'явиться повідомлення про те, що в базі даних таких користувачів не знайдено. Аналогічно відбувається і процес пошуку студентів порталу за допомогою меню «Пошук студентів».

Рисунок 3.7 – Пошук юзерів

Далі є меню для введення нових курсів порталу, після натискання на нього відкриється сторінка з вікном для введення даних нового курсу (рисунок 3.8). Тут потрібно ввести найменування курсу, П.І.Б. тьютора і позначити курс - якщо мітка «Активність» відзначена, то курс буде доступний для студентів, в іншому випадку - не доступний.

Рисунок 3.8 Вікно для створення нового курсу

Інформацію про всі курсах можна переглянути за допомогою посилання «Курси», після натискання на яку з'явиться сторінка зі списком наявних в порталі курсів (рисунок 3.9). Тут як і в випадку з операторами є можливість вибіркового видалення курсів. Причому при видаленні будуть видалені як сам курс, так і вся інформація пов'язана з ним: питання, оцінки студентів по курсу, і т.д.

Рисунок 3.9 - Сторінка зі списком курсів порталу

На цій сторінці за допомогою натискання мишею на назву курсу з'явиться сторінка з детальною інформацією по обраному курсу (рисунок 3.11). Тут є можливість зміни всієї інформації про курс, зокрема найменування, тьютора, активність. Крім того тут же можна видалити поточний курс (посилання «Видалити»). Далі бачимо інформацію про ліміти курсу - максимальна кількість перездач тих чи інших контролів і кількість питань при проходженні контролів - тут же ліміти можна редагувати. Причому якщо ліміти на кількість питань не проставлені, то з'являється

відповідне текстове попередження про неможливість проходження контролів за курсом студентами. За замовчуванням ліміти порожні і потребують редагування.

Так само на цій сторінці є можливість перегляду успішності студентів з даного курсу і додавання нових питань.

Рис. 3.10 - Сторінка з детальною інформацією про курс

Якщо з даного курсу в базі даних є питання, то буде виведена інформація про питаннях курсу в розрізі контролів, подібна наступною:

Рисунок 3.11 - Інформація про питання курсу

Тут меню «Питання курсу» переводить адміністратора на сторінку з питаннями курсу, подібну зображеної на наступному малюнку:

Рисунок 3.12 - Сторінка з питаннями курсу

Кожне питання можна переглянути детально, або видалити. Причому висновок питань відбувається посторінково, якщо загальна кількість питань курсу перевищує 15, то наступна порція питань буде відображена на наступній сторінці і т.д.

Введення нових питань курсу відбувається за допомогою меню «Додати питання», після натискання на яке з'явиться вікно наступного вигляду:

Рисунок 3.13 Форма для введення нового питання

Тут потрібно ввести текст питання, чотири варіанти відповіді на нього, правильну відповідь, і контроль до якого буде ставитися дане питання. Після чого після натискання на кнопку «ОК» питання буде негайно доданий в базу даних.

Крім того в самому низу сторінки з інформацією по курсу можна помітити інтерфейс для редагування доступу до курсу:

Рисунок 3.14 - Інтерфейс редагування доступу до курсу

Тут адміністратор може змінювати доступ операторів до поточним курсом, причому доступ можна редагувати як вибірково, так і всім операторам відразу. Галочка у відповідній колонці означає, що оператор має доступ до дії, описаного в заголовку колонки, і навпаки.

Меню «Питання» у верхній частині інтерфейсу адміністратора переводить його на сторінку зі списком курсів порталу та інформацією про їх питаннях:

Рис. 3.15. Сторінка «Питання»

Тут меню «Перегляд» переводить адміністратора на сторінку подібну зображеної на рисунку 3.15. Причому дане меню з'являється тільки навпроти тих курсів, які мають питання в базі даних. Так само навпроти назв таких курсів з'явиться кількість всіх питань курсу.

Меню «Пошук питань» відкриє форму введення параметрів для пошуку запитань. Тут треба просто ввести ключове (і) слово (а), яке (і) може (уть) міститися або в тексті питання, або в його відповідях.

Ну і нарешті, меню «Вихід» забезпечує вихід адміністратора з системи, після

чого користувачеві знову відобразиться вікно авторизації адміністратора.

### 3.2 Оператор-модуль

Для роботи з операторським блоком необхідно ввести в адресний рядок браузера *“http: //localhost/test_portal/oper/oper.php”*. Після чого перед нами з'явиться вікно авторизації для входу в оператор-модуль, ідентичне зображеному на малюнку 3.16. Після успішної авторизації перед користувачем відкриється вікно інтерфейсу оператора:

Рисунок 3.16 - Головне вікно інтерфейсу оператора

Тут є меню «Персональні дані», «Зміна пароля», «Доступ» і «Вихід». Перші два з них ідентичні однойменним пунктам меню в адмін-модулі і процес роботи з ними точно такий же. Більш цікавим є пункт «Доступ», через який ведеться вся робота оператора з даними порталу. При натисканні на посилання «Доступ» відкриється сторінка зі списком усіх курсів, до яких оператор має хоча б мінімальний доступ (доступ на читання):

Рисунок 3.17 - Сторінка «Доступ» оператор-модуля

Тут після натискання на назву конкретного курсу оператор буде переведений на сторінку для роботи з обраним курсом, подібну зображеної на малюнку 3.17. Дані на такій сторінці будуть відображені відповідно до доступом оператора до даного курсу. При мінімальному доступі буде відображена наступна інформація - найменування і тьютор курсу, інформація про питаннях курсу в розрізі контролів і безпосередньо список питань курсу з можливістю детального перегляду кожного питання. На наступній сторінці зображена сторінка роботи з курсом в разі коли оператор крім мінімального доступу має доступ до лімітів курсу і на вставку нових питань. Тут

робота з лімітами і додаванням питань ідентична з адмін-модулем. При натисканні на посилання «Докладно» наявну в кожному питанні відкриється сторінка з детальною інформацією про питання (рисунок 3.15). Причому як бачимо якщо оператор не має доступ на зміну, то у вікні питання чи не з'явиться кнопка «ОК» для зміни даних питання. Точно так же - якщо оператор не має доступу на видалення, то посилання видалення питання не буде з'являтися, в іншому випадку вона буде праворуч від посилання «Докладно»:

Рисунок 3.18 Оператор має доступ на видалення питань

Рисунок 3.19 - Сторінка для роботи з обраним курсом

Рисунок 3.20 - Форма для перегляду питання (оператор не має доступу на зміну)

Меню «Вихід» працює практично аналогічно однойменним меню в адмін-модулі і переводить користувача на сторінку авторизації оператора.

### 3.3 Студент-модуль

Для роботи зі студентською блоком необхідно ввести в адресний рядок браузера http: //localhost/test_portal/index.php. Після чого перед нами з'явиться вікно авторизації для входу в студент-модуль (рисунок 3.21). Тут як бачимо, крім форми авторизації є посилання «Реєстрація». Це посилання потрібна для реєстрації нових студентів, після натискання на яку відкриється сторінка реєстрації (рисунок 3.22).

Рисунок 3.21 - Вікно авторизації студентів

Тут нам потрібно ввести П.І.Б., дату народження, e-mail, логін і пароль. Причому знову ж як і у випадку з операторами перевіряється - чи немає в базі студентів тільки що введеним логіном. Після успішної реєстрації користувач отримує можливість увійти в особистий кабінет використовуючи тільки що введені логін і пароль.

Рисунок 3.22. Форма для реєстрації нового студента

Після успішної авторизації студент буде переведений на головну сторінку особистого кабінету, подібну до наступного:

Рисунок 3.23. Головне меню студент-модуля

Тут є меню «Персональні дані», «Зміна пароля», «Курси» та «заліковку». Перші два з них ідентичні однойменним пунктам меню в адмін- і оператор-модулях і процес роботи з ними точно такий же. Більш цікавим є пункт «Курси», через який ведеться вся робота студента з порталом. При натисканні на посилання «Курси» перед студентом з'явиться список всіх доступних курсів порталу:

Рисунок 3.24 - Сторінка «Курси» студент-модуля

Після чого студент може вибрати конкретний курс і пройти контролі знань з нього. З'явиться сторінка зі статусом студента за обраним курсом:

Рисунок 3.25 - Вікно статусу студента за обраним курсом

Як бачимо при першому взаємодії з курсом студент має право пройти тільки перший поточний контроль (тільки він має посилання на проходження контролю). Тут будуть відображатися набрані бали за контролям, кількість перездач студента по кожному контролю, ліміти на перездачу кожного контролю (щоб студент мав відомості про те, скільки разів він може перездати той чи інший контроль) і дати останніх здач (перездач) контролів. Натиснувши на посилання конкретного контролю студент буде переміщений на сторінку безпосереднього проходження обраного контролю, де будуть відображені питання контролю відповідно до настройками зазначеними операторами або адміном (рисунок 4.4.28). Якщо обраний контроль містить ліміт на перездачу рівний нулю, тобто перездача забороняється (контроль можна пройти всього один раз), тоді студенту видається наступне оповіщення:

Рисунок 3.26 - Попередження про неможливість перездачі контролю

Після проходження чергового контролю студенту буде доступне посилання на наступний контроль, і т.д. При натисканні на кнопку «Підтвердити» студенту буде видано результат здачі контролю в наступному вигляді(рисунок 3.27):

Рисунок 3.27 Один з можливих результатів здачі контролю

Риунок 3.28 – Вигляд тестування

Меню «заліковка» дозволяє студенту побачити оцінки з усіх курсів порталу в

зручному вигляді в розрізі контролів:

Рисунок 3.29 - Сторінка «заліковка» студентського-модуля

Звідси ж він може знову перейти на сторінку свого статусу по обраному курсу, натиснувши на назву курсу, а меню «Вихід» працює аналогічно однойменним пунктам в двох попередніх модулях.

# ВИСНОВОК

Отже, в даній роботі були викладені основні відомості про стан сучасного дистанційного навчання як у всьому світі, так і в Україні зокрема. Були дані поняття як систем дистанційного навчання в цілому, так і відкритого дистанційного навчання. З'ясовано, що відкрите дистанційне навчання - це кілька видозмінений тип традиційного дистанційного навчання, який дає можливість всім бажаючим отримати ті чи інші знання без «тимчасових» обмежень, незалежно від їх географічного розташування, національності, соціального статусу. Причому відкрите навчання не має на увазі видачу дипломів, хіба що у вигляді рідкісних винятків. Відкрите дистанційне навчання не покликане навчати нової спеціальності, але дає можливість зручної і швидкої перепідготовки кадрів, підвищення кваліфікації без відриву учнів від основної діяльності. Визначено, що основна вимога до відкритих систем навчання - це можливість навчаються самим планувати свій графік навчання,

Встановлено, що з усіх видів контролю знань у відкритому дистанційному навчанні найбільш ефективним і прийнятним є тестовий контроль. Такий контроль здатний забезпечувати режим реального часу, моментальну видачу результатів, різноманітність і випадковість завдань, не вимагає підвищених витрат на спеціалізовану апаратуру і високошвидкісний інтернет, а також не обмежує навчаються в часі. Виявлено також і недоліки тестового контролю, серед яких основним є втрата ідентифікації особистості контрольованого, в наслідок чого були наведені приклади зменшення ступеня студентського шахрайства як педагогічними, так і програмними засобами.

В основній частині роботи детально було розглянуто приклад тестового порталу контролю знань учнів, який може використовуватися в системах відкритого дистанційного навчання. Висвітлено основні моменти програмної частини порталу та особливості програмного контролю знань учнів, а також управління порталом з боку

організаторів дистанційного навчання.. Тестовий портал - це всього лише мала частина реальної системи дистанційного навчання. Звичайно ж при створенні такої системи контроль знань не повинен бути винесений в окремий модуль, а повинен бути тісно взаємопов'язаний з самою системою, зокрема з лекціями курсів системи навчання.