МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ОДЕСЬКИЙ ДЕРЖАВНИЙ ЕКОЛОГІЧНИЙ УНІВЕРСИТЕТ

Факультет магістерської та
аспірантської підготовки
Кафедра інформаційних техноло-
гій

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему: Розробка підсистеми планування проведення та оцінювання знань

Виконав студент 2 курсу групи МК-61
спеціальності 8.05010101 Інформаційні
управляючі системи та технології,
Хаджиу Едуард Михайлович

Керівник к.т.н., доцент,
Гнатовська Ганна Арнольдівна

Рецензентк.геогр. н., доцент,
КузніченкоСвітланаДмитрівна

Одеса 2017

АНОТАЦІЯ

Тема магістерської роботи**“**Розробка підсистеми планування проведення та оцінювання знань”.

Актуальність магістерської роботи полягає в необхідності використання сучасних можливостей мережі Internet, web-сервісів, які стосуються надання репетиторської допомоги студентам та проходження перевірки отриманих знань шляхом проходження тестування у підсистемі планування та оцінювання знань.

Об'єкт дослідження –процеси розробки підсистеми для проходження перевірки знань.

Мета роботи – розробка підсистеми, яка допоможе визначити якість знань після проходження віртуальнго курсу навчання студентом та яка покаже рівень засвоєння пройденого матеріалу у системі віртуального навчального центру.

Для реалізації поставленої мети були вирішені наступні питання: проведено дослідження призначення тестування і особливостей підсистеми планування та оцінювання знань, проведен аналіз та дослідження видів та самої мети тестування; виявили класифікацію тестування; визначені формати тестових питань; обрана архітектура та програмні засоби реалізаціїweb-сервісу; здійснена програмна реалізація. Практична цінність магістерської роботи полягає в тому, що розроблену підсистему зручно використовувати для перевірки знань шляхом проходження онлайн тестування.

Ключові слова: інформаційна підсистема, проходження тестування, користувач-викладач, проектування.

Магістерська робота містить в собі 90 сторінок, 21 таблицю, 37 рисунків, 14 посилань та 10 листів додатків.

ANNOTATION

The topic of master work "The development of subsystem for planning and assessment of knowledges".

The relevance of this master work is the need to use advanced network capabilities Internet, web-services, which include providing assistance to students and passing checks acquired knowledge by testing the subsystem in planning and assessment.

Object of research – processes of the subsystem for passing assessment.

Purpose – development subsystem that can help determine the quality of knowledge of student after passing the online course and that will show the level of mastering the material covered in the system virtual training center.

To achieve this goal have been resolved following issues: the research purpose of testing and subsystem features the planning and assessment, the analysis and research of species and the very purpose of testing; found a classification of testing; defined formats of test questions; chosen software architecture and implementation of web-service; made software implementation. Practical value of master's thesis is that the developed subsystem is useful to test student knowledge by passing the online test.

Keywords: information subsystem, testing, user-teacher, design.

ЗМІСТ

ВСТУП 12
1 РОЛЬ ТЕСТУВАННЯ У ПРОЦЕСІ КОНТРОЛЮ ЗА НАВЧАННЯМ 13
1.1 Дослідження призначення тестування і особливостей підсистеми планування та оцінювання знань 13
1.2 Визначення мети тестування 14
1.3 Класифікація тестів 15
2 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАВДАННЯ 25
2.1 Дослідження предмета, цілей і особливостей підсистеми планування та оцінювання знань 25
2.2 Дослідження предметної області 26
2.3 Перелік термінів, визначень і скорочень 28
3 ВИБІР І ОБГРУНТУВАННЯ АРХІТЕКТУРИ СИСТЕМИ ТА ПРОГРАМНИХ ЗАСОБІВ РЕАЛІЗАЦІЇ 30
3.2 Вибір HTTP сервера 31
3.2.1 Сервер Apache 32
3.2.2 Сервер Nginx 34
3.3 Опис схеми співпраці двох HTTP-серверів 35
3.4 Вибір системи управління базами даних 36
3.5 Вибір мови програмування 37
3.6 Вибір PHP Framework 42
3.7 Система управління версіями GIT 43
4 ПРОЕКТУВАННЯ ІНФОРМАЦІЙНОЇ ПІДСИСТЕМИ 45
4.1 Проектування інформаційної підсистеми оцінювання знань за допомогою методології функціонального моделювання SADT 45
4.2 Проектування функціональних вимог і БД системи 53
4.3 Опис процесів підсистеми планування проведення та оцінювання знань 61
4.4 Проектування служб для користувача інтерфейсу 64
5 ПРАКТИЧНА РЕАЛІЗАЦІЯ ЗАСТОСУВАННЯ КОРИСТУВАЧА 68
5.1 Керівництво користувача 68
5.2 Керівництво користувача-викладача 71
5.2.1 Створення питань для тестування 75

5.2.2 Проходження тестування у підсистемі планування та оцінювання знань..... 80
ВИСНОВКИ..... 82
ПЕРЕЛІК ПОСИЛАНЬ..... 83
ДОДАТОК А Основні коди програмних модулів ..... **Ошибка! Закладка не определена.**

ПЕРЕЛІКСКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ І ТЕРМІНІВ

Скорочення

CSS – Cascading Style Sheets – каскадні таблиці стилів

HTML – HyperText Markup Language – мова гіпертекстової розмітки

PHP – Hypertext Preprocessor – «PHP: препроцесоргіпертекста»

Терміни

Аккаунт –обліковий запис, де зберігається персональна інформація користувача для входу на сайт

Хостинг – послуга з надання простору для розміщення сайтів в мережі Інтернет

Apache – вільний веб-сервер

MySQL – система управління базами даних (СУБД).

CloudFlare – програма для захісту від Ddos-атак

ВСТУП

Ні для кого вже не новина, що комп'ютери стали нашими помічниками у всьому, в тому числі і в отриманні освіти. Зараз навіть є онлайнові школи і курси, де учні взагалі не відвідують навчальний заклад, а вчаться повністю за допомогою мережі. В даний час різні перевірки знань проводять у вигляді тестів, реалізованих на комп'ютері. Ви можете знаходитись у іншому місці або країні, та проходити навчання через мережу Internet. Такі системи вже давно використовуються західними вищими або середніми закладами навчання.

На сьогоднішній день тестування глибоко проникло в багато сфер діяльності людства. В освіту, роботу, медицину і т.д. Тести особливо популярні в Америці, вони служать для відбору найбільш гідних кандидатів. Багато бізнесменів вважають цю систему дуже зручною, так як вона дозволяє малими зусиллями вирішити проблему комплектації кадрів. Фахівці вважають, що тести більш надійні в прогнозі несприятливих результатів (наприклад, вам не варто займатися певним видом діяльності). А ось розвиненість будь-яких якостей або здібностей визначається з меншим ступенем вірогідності. Тому за допомогою тестового відбору легше обмежити коло претендентів. У багатьох країнах світу вищі та середні навчальні заклади також активно використовують тестування для перевірки та оцінювання знань студентів або учнів.

Актуальність даної магістерської роботи полягає в необхідності використання сучасних можливостей інформаційних технологій та технологій мережі Internet в процесі навчання. Розробка підсистеми планування проведення та оцінювання знань дозволяє здійснити якісну перевірку знань отриманих при проходженні курсу навчання у віртуальному навчальному центрі. Тобто дана підсистема є доповненням системи віртуального навчального центру розробленої мною у дипломному проекті бакалавра.Ця інформаційна система призначена для підвищення ефективності управління та контролю над навчальним процесом, підвищення прозорості та об'єктивності оцінювання учнів, а також для зручності здійснення процесу навчання з застосуванням сучасних інформаційних технологій надання, доступу та обміну інформацією. Розроблена підсистема допоможе визначити якість знань після проходженні віртуального курсу навчання студентом та покаже рівень засвоєння пройденого матеріалу.

1 РОЛЬ ТЕСТУВАННЯ У ПРОЦЕСІ КОНТРОЛЮ ЗА НАВЧАННЯМ

1.1 Дослідження призначення тестування і особливостей підсистеми планування та оцінювання знань

З давніх часів тестування було одним з найбільш зручних способів перевірки і оцінки знань. Тестування – від англ. “test”– випробування, перевірка. Саме слово «тест» має французьке коріння і означає посудину, яке використовується в аптечній справі для проведення різних дослідів. У сучасному контексті це поняття вживають, коли треба пройти перевірку, пробу, випробування. Багато людей звикли, що тестові завдання представлені у вигляді друкованих питань з варіантами відповідей. Однак прабатьками сучасних тестів були певні випробування, які пропонувалися претенденту на посаду в якості перевірки [1]. Так, ще близько двох тисяч років тому в Китаї кандидатів на урядові посади відбирали за допомогою тестової перевірки, а в Стародавньому Єгипті посвячення в жерці відбувалося після серії випробувань.

На сьогоднішній день тестування глибоко проникло в багато сфер діяльності людства. В освіту, роботу, медицину і т.д. Тести особливо популярні в Америці, вони служать для відбору найбільш гідних кандидатів. Багато бізнесменів вважають цю систему дуже зручною, так як вона дозволяє малими зусиллями вирішити проблему комплектації кадрів. Фахівці вважають, що тести більш надійні в прогнозі несприятливих результатів (наприклад, вам не варто займатися певним видом діяльності). А ось розвиненість будь-яких якостей або здібностей визначається з меншим ступенем вірогідності. Тому за допомогою тестового відбору легше обмежити коло претендентів.

Тест – це інструмент, що складається з системи тестових завдань, стандартизованої процедури проведення і заздалегідь спроектованої обробки і аналізу результатів, призначений для вимірювання якостей і властивостей особистості, зміна яких можливо в процесі систематичного навчання.

Тести як вимірювальний інструмент використовуються в більшості країн світу. Їх розробка і використання засновані на потужній теорії і підтверджені численними емпіричними дослідженнями. Тестологія як теорія і практика тестування існує більше 120 років, і за цей час накопичено величезний досвід використання тестів у різних сферах людської діяльності, включаючи освіту[2]. Тести не є універсальним засобом, межі їх використання добре відомі. Все це створює упевненість в тому, що якісно підготовлений

і використаний тестовий інструмент дасть якісну і надійну інформацію, відповідну реальному стану справ.

Тести – не тільки якісний, але і об'єктивний спосіб оцінювання. Об'єктивність тестування досягається шляхом стандартизації процедури проведення (на всіх етапах проведення тестування неможливо внести суб'єктивну складову в оцінку) і шляхом стандартизації та перевірки показників якості окремих завдань і тестів цілком.

Привабливими виявляються тести і з точки зору управління. Вони дають широку можливість для варіювання складності тестового матеріалу, широти охоплення, цільової спрямованості, включення в тест декількох компонентів структури знань, що дозволяє створити інструмент, що враховує найвибагливіші вимоги управлінця. Система показників якості тесту дає можливість оцінити, наскільки створений інструмент реально відповідає цим вимогам, і використовувати його в суворій відповідності з ними.

Крім цього, тести – ефективний інструмент з економічної точки зору. При тестуванні основні витрати припадають на складання якісного інструментарію, тобто носять разовий характер. При збільшенні кількості атестуються ці витрати розподіляються на них пропорційно, що призводить до зниження загальних витрат.

Приступаючи до детального розгляду ролі тестування в процесі контролю знань, спочатку необхідно визначити, що відрізняє тести від інших способів перевірки знань учнів. Формально це зробити досить просто, використовуючи наведене визначення: тест повинен мати в якості складових, принаймні, три елементи – систему завдань, зафіксовану документально технологію пред'явлення і відпрацьовану систему перевірки обробки і аналізу рзультатів, які повинні складати єдність.

1.2 Визначення мети тестування

В цьому розділі обговорюється залежність розробки тестів від тих цілей, які стоять перед розробниками систем моніторингу.

Перш за все, потрібно зупинитися на двох підходах, які в даний час склалися в тестуванні – тестах, орієнтованих на критерій (критеріальною-орієнтованих), і тестах, орієнтованих на норму (нормативно-орієнтованих). З'явилися як різні підходи до аналізу результатів тестування, що відображають різні підстави для порівняння, зараз ці два підходи визначають різницю на всіх етапах створення тесту.

У найзагальнішому вигляді підставою для порівняння в тестах, орієнтується на норму, є результати, отримані при попередньому тестуванні групи учнів, репрезентативною для якоїсь спільності. Оцінка в рамках цього підходу дається на основі попередньо отриманих статистично обґрунтованих норм.

Характеризуючи підхід, орієнтований на критерій, необхідно підкреслити, що результати другого типу тестів обробляються з точки зору спеціальних знань або навичок, які студент може продемонструвати. Він дає можливість визначити, що кожен студент може зробити з точки зору конкретного завдання, що не співвідносячи його дії з діями інших членів групи. Цей підхід дає оцінку тільки по дихотомічної шкалою: впорався – не впорався, пройшов – не пройшов, залік – незалік і т.д. Однак він має широкі можливості для опису тих завдань, з якими учень справляється індивідуально, тих завдань, з якими справляється найменше учнів. Така орієнтація дає можливість реалізувати великі діагностичні можливості цього підходу.

1.3 Класифікація тестів

Тести класифікуються за різними ознаками. По виду властивостей особистості вони поділяються на тести досягнень і особистісні. До перших відносяться тести інтелекту, шкільної успішності, тести на творчість, тести на здібності, сенсорні і моторні тести. До других – тести на установки, на інтереси, на темперамент, характерологічні тести, мотиваційні тести. Однак не всі тести (наприклад, тести розвитку, графічні тести) можна впорядкувати за цією ознакою. За способом застосування розрізняються індивідуальні та групові тести. При груповому тестуванні одночасно обстежується група впробовуваних. Якщо в тестах рівня тимчасових обмежень немає, то в тестах на швидкість вони обов'язкові. Залежно від того, наскільки в результаті тстування виявляється суб'єктивність дослідника, розрізняють тести об'єктивні і суб'єктивні.

До об'єктивних тестів відноситься більшість тестів досягнень і психофізіологічні тести, до суб'єктивних – проектні тести. Цей поділ певною мірою збігається з поділом на прямі і непрямі тести, які різняться в залежності від того, знають або не знають випробувані значення та мету тесту.

Для проектних тестів типова ситуація, коли випробуваний не поінформований про дійсної мети дослідження. При виконанні завдань про-

ективних тестів не існує "правильних" відповідей. Залежно від наявності в тесті мовного компонента розрізняються тести вербальні і невербальні. Вербальним, наприклад, є тест на словниковий запас, невербальних – тест, що вимагає як відповідь певних дій.

За формальній структурі розрізняються тести прості, тобто елементарні, результатом яких може бути єдина відповідь, і тести складні, що складаються з окремих підтестів, по кожному з яких повинна бути дана оцінка. При цьому можуть вираховуватися і загальні оцінки.Кожне тестування має свої плюси і мінуси.

Переваги тестування:

- можливість проводити масові вимірювання знань;
- можливість встановити рівень знань учня по предмету в цілому і по окремих його розділів;
- тест це більш точний інструмент, так, наприклад, шкала оцінювання тесту з 20 питань, складається з 20 ділень, в той час, як звичайна шкала оцінки знань – тільки з чотирьох;
- оперативність і економічність;
- всі учні котрі тестуються знаходяться в однакових умовах;
- об'єктивність оцінки знань.

Недоліки тестування:

- скрізь різний рівень знань;
- можливість вгадування відповідей;
- дані, одержувані викладачем в результаті тестування, хоча і включають в себе інформацію про прогалини в знаннях по конкретних розділах, але не дозволяють судити про причини цих прогалин;
- розробка якісного тесту – тривалий, трудомісткий і дорогий процес. Стандартні набори тестів для більшості дисциплін ще не розроблені, а розроблені зазвичай мають дуже низьку якість;
- тест не дозволяє перевіряти і оцінювати високі, продуктивні рівні знань, пов'язані з творчістю;
- широта охоплення тем в тестуванні має і зворотну сторону. Учень при тестуванні, на відміну від усного або письмового іспиту, не має достатньо часу для скільки-небудь глибокого аналізу теми.

У наш час більшість навчальних закладів та вищих навчальних закладів використовують таку форму тестування, як інтернет-тестування.Також тести можна класифікувати на наступним підставах:

За процедурою створення можуть бути виділені стандартизовані і нестандартизовані тести.

Стандартизируються процедура і умови проведення тестування, сособи обробки і інтерпретації результатів, які повинні привести до створення рівних умов для випробовуваних і мінімізувати випадкові помилки і похибки як на етапі проведення, так і на етапі обробки результатів та інтерпретації даних.

В освіті можна виділити ряд завдань, які можуть бути вирішені нестандартизованими тестами. Однак для цілей моніторингу необхідно використовувати тільки стандартизований тестовий інструмент.

За засобами представлення виділяються тести:

– з використанням тестових зошитів, в яких знаходяться тестові завдання і в яких випробовуваний фіксує результати;
– бланкові, коли випробовувані відзначають або вписують правильні відповіді (фіксують відповіді) на спеціальних бланках. Бланки заявляються окремо від завдань;
– комп'ютерні.

Для моніторингу підходить будь-який з цих способів, але потрібно пам'ятати про одне – пред'являючи один і той же тест в різних формах, ми отримаємо різні результати. Не можна порівнювати результати тестування, отримані при різних способах пред'явлення.

За провідною орієнтації виділяються:

– тести швидкості. Вони містять прості завдання, але час вирішення обмежена настільки, що жоден випробуваний не встигає вирішити за відведений час всі завдання;
– тести потужності або результативності включають важкі завдання. Час їх вирішення або зовсім не обмежена, або м'яко лімітовано. Оцінці підлягають успішність і спосіб вирішення завдання. Прикладом такого роду тестових завдань можуть бути завдання для письмових підсумкових іспитів за курс основної школи;
– змішані тести, які об'єднують в собі риси двох перерахованих вище. У них представлені завдання різного рівня складності, від найпростіших до дуже складних. Час випробування в даному випадку обмежена, але досить для вирішення більшості пропонованих завдань більшістю обстежуваних.

Оцінкою в даному випадку служать як швидкість виконання завдань (кількість виконаних завдань), так і правильність рішення. Ці тести найбільш часто застосовуються на практиці і саме до них відноситься більшість тестів навчальних досягнень, які можна використовувати для потреб моніторингу.

За видом нормування виділяють:

– тести, орієнтовані на статистичні норми. Підставою для порівняння в них служать відповідним чином обґрунтовані, статистично отримані значення виконання даного тесту репрезентативною вибіркою досліджуваних;
– критеріальною-орієнтовані тести. Вони призначені для визначення рівня індивідуальних досягнень випробуваного щодо заданого критерію, існуючого в реальній практиці і заздалегідь відомого: рівня знань, умінь, навичок, необхідних для будь-якого виду діяльності;
– ненормовані.

За цілями використання виділяються наступні групи тестів:

– знань або поведінки студента на початку навчання (визначає тест);
– прогресу, досягнутого в процесі навчання (формуючий тест);
– труднощів навчання і їх джерел під час процесу навчання (діагностичний тест);
– основних досягнень в кінці навчання (суммирующий тест).

Принципи та механізми розробки однакові для всіх видів цих тестів, але вміст матеріалу, включеного в тест, і ступінь складності питань повинні відповідати цілям тестування.

Попередній визначає тест призначений для оцінки початкових здібностей, зазвичай є нескладним і охоплює дуже невеликий діапазон знань. Він може зачіпати мінімум базових знань по темі навчання або інший обмежений набір необхідних знань і практично не відрізняється від підсумовує тесту, що дається в кінці курсу (розділу) навчання.

Яка Формує тест, який використовується для контролю над прогресом навчання, зачіпає обмежений сегмент навчання. З його допомогою робиться спроба оцінити всі важливі результати вивчення даного сегмента. Акцент робиться на оцінці ступеня володіння матеріалом і забезпечення зворотного зв'язку зі студентом щодо коригування окремих помилок в тих областях, де він не досяг успіхів. Таким чином, формує тест складається з серії окремих тестових питань, всебічно охоплюють обмежену область навчання. Він розробляється так, щоб дати учневі конкретні інструкції для виправлення виявлених в результаті тесту помилок. Дані тесту є навчальними, вони зазвичай менш складні, ніж підсумовують тести, що даються в кінці процесу навчання.

Діагностичний тест містить відносно велику кількість питань, що мають відношення до конкретної тестованої області. Оскільки метою тесту є визначення труднощів навчання, увага зосереджується на відповідях учнів на

конкретне питання або групу питань, загальний бал має другорядне значення. Цей тест зазвичай більше фокусується на поширених помилках учнів, ніж на спробі широкого відбору очікуваних результатів навчання. Тести даного тпу розроблені для учнів, у яких є проблеми в навчанні, і мають дуже нисокий рівень складності.

Суммирующий тест розробляється для оцінки широкого діапазону результатів навчання, очікуваного в кінці навчального процесу. Складність і представництво вибірки є важливими аспектами даного тесту, так як його рзультати використовуються для проставляння балів і визначення ступеня дсягнення завдань курсу навчання. Для того щоб адекватно відібрати всі очікувані результати навчання, що підсумовує тест зазвичай містить питання більш високого рівня складності, ніж інші види тестів.

Для потреб моніторингу можна використовувати три з наведених видів тестів.

1.4 Формати тестових завдань (ТЗ)

Хороше тестове питання повинно задовольняти два основних критерій. По-перше, тестове питання має бути важливим за змістом. Безумовно, зміст тестових завдань є дуже важливим, але концентруватися тільки на ньому недостатньо. По-друге, тестові завдання, покликані оцінювати критично важливі знання, які не будуть виконувати свого призначення, поки вони не будуть мати хорошу структуру. Необхідно уникати помилок, від яких виграють досвідчені в тестуванні школярі і студенти, а також уникати надмірної складності. Це передумови того, що тестові завдання будуть давати валідні результати.

Все розмаїття питань множинного вибору можна розділити на дві великі категорії тестових завдань:

- вимагають від екзаменованих вибрати всі підходящі відповіді;
- вимагають від екзаменованих вказати один відповідь (один правильнийвідповідь).

Кожна категорія представлена кількома специфічнимиформатами, зазначеними нижче.

Формат А. Тестові завдання формату "З відповідями на вибір". Це найбільш поширений формат тестового завдання. Він передбачає наявність одного питання і декількох варіантів відповідей, один з яких є правильним. Формат А "З відповідями на вибір" – досить гнучкий формат, що, мабуть, і визначає його поширеність. Дистрактори в ньому можуть бути або абсолют-

но неправильними, або правильними лише частково. Відповіддю на завдання вважається найбільш правильний з усіх дистракторов. Імовірність вгадування правильної відповіді для цього формату найбільша серед усіх форматів тестових завдань, тому не рекомендується використовувати кількість варіантів відповідей менше або більше чотирьох.

Наприклад: Основна функція тромбоцитів – це ...

a) Гемостаз крові
b) Освіта онкотичного тиску
c) Забезпечення імунітету
d) Транспортіровка CO2 і O2

Формат B. Формат B передбачає наявність одного питання і декількох варіантів відповідей, частина яких (зазвичай більше одного) є правильними. Формат B накладає жорсткі умови на дистрактори. На відміну від формату A дистрактори повинні бути абсолютно неправильними відповідями. Оскільки кількість правильних відповідей невідомо, то частково правильні дистрактори завжди будуть викликати суперечки, чи є вони правильними відповідями чи ні. Кількість варіантів відповідей, як правило, не менше 4. Імовірність вгадування правильної відповіді значно менше, ніж у формату A, і її можна не враховувати при обробці.

Наприклад: Виберіть нижчезазначених персько-таджицьких класиків.

a) А. Рудакі
b) А. Фірдавс
c) А. Лохуті
d) А. Джамі

Формат C. Тестове завдання на відповідності. Формат C – це формат, який об'єднує завдання, які можна представити у вигляді логічних пар. Формат передбачає наявність в завданні питання і декількох тверджень, до яких необхідно підібрати логічні відповідності з деякого набору. Набір відповідностей, як правило, перевищує кількість тверджень. Наприклад: У лабораторії фізики вимірюють ряд фізичних величин, які наведені в лівому стовпчику. Установіть відповідність між вимірами, проведеними школярем, і приладами, за допомогою яких він вимірював значення даних величин(рис.1.1).

Величины

A.Сопротивление
B.Мощность
C.Сила тока
D.Напряжение

Приборы

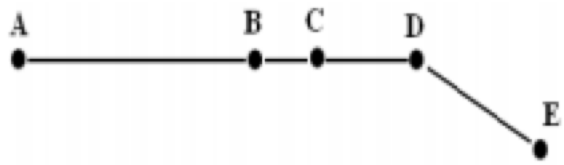
1.Ваттметр
2.Омметр
3.Электрометр
4.Амперметр
5.Вольтметр

Ответ:

	1	2	3	4	5
A		x			
B	x				
C				x	
D					x

Рисунок 1.1 – Приклад тестового питання формата C

Формат D. Формат D – це формат, в якому порядок тверджень має значення. Як правило, питання тестового завдання цього формату вимагає розташувати по порядку набір тверджень, що описують будь-які дії (тхнологічний процес), або впорядкувати затвердження по будь-якою ознакою. Наприклад: На малюнку представлена траєкторія руху автомобіля, яка проходить зазначені ділянки за однаковий час. Встановити послідовність значень швидкості автомобіля в даних ділянках по зменшенням(рис.1.2).



Ответ :

A	AB
B	DE
C	CD
D	BC

Рисунок 1.2 – Приклад тестового питання формата D

Формат E. Формат E – це відкритий формат, який передбачає наявність питання і поля введення відповіді студента. Студент повинен ввести в поле введення свою відповідь в довільній формі. Відкритий формат є потужним засобом перевірки знань, оскільки студент повинен самостійно, без наведених варіантів відповідей (які все-таки є в певній ступеня підказкою), правильно відповісти на питання завдання. Наприклад: Тролейбус рухається зі швидкістю 36 км/год. На якому відстані від зупинки водій повинен почати гальмувати, повідомляючи тролейбусу прискорення не більше 1 м/с2?(рис.1.3).

Рисунок 1.3 – Приклад правильної відповіді у тестовому питанні формату D

1.5 Планування тестових питань, різні види тестів

Відома в педагогіці методика оцінки знань студентів шляхом усної перевірки не тільки застаріла, але найчастіше дає лише якісне уявлення про рівень засвоєння ними певної суми знань і умінь. В умовах індивідуалізації навчання, впровадження у вищих навчальних закладах кредитно-модульної системи організації навчального процесу з упором на самостійну роботу студентів надійно обробляти результати навчальної діяльності дозволяють тільки методи письмовій або практичної перевірки, що дають кількісну оцінку рівня навчальних досягнень. Це дозволяє виключити прояв суб'єктивності при формуванні оцінки та її залежність від особистості викладача і його кваліфікації.

Останнім часом у педагогічній практиці все частіше використовуються методи програмованого контролю знань, засновані на єдиній і об'єктивній методиці аналізу сформованої діяльності. В основу покладено виявлення рівня засвоєння за допомогою спеціально розроблених завдань (тестів) практичного і теоретичного характеру, адекватних сформованій на даному рівні діяльності.

Характеристика діяльності може бути представлена трьома послідовними рівнями:

- I рівень (рівень знайомства з предметом) – впізнавання (впізнання), розрізнення, класифікація об'єктів, систем, властивостей; аналіз і відтворення результатів можливих дій на основі досвіду, знань, пам'яті;
- II рівень (рівень умінь) – продуктивна діяльність по раніше засвоєного зразком на деякій множині об'єктів;
- III рівень (рівень «творчості») – продуктивна діяльність на безлічі об'єктів шляхом самостійного конструювання програми діяльності.

Пропонована методика програмованого контролю знань заснована на єдиній і об'єктивній методиці аналізу сформованої діяльності, інваріантному змісту навчальних дисциплін, може бути використана як при підсумковому, так і проміжному контролі знань. В основу покладено виявлення рівня засвоєння за допомогою спеціально розроблених контрольних завдань

(тестів) практичного і теоретичного характеру, адекватних сформованій на даному рівні діяльності. Про якість виконаної діяльності можна судити по логічним і істотним операціями тесту (одне або кілька взаємопов'язаних дій), необхідних для його рішення.

При використанні набору тестів різних рівнів (каскадний тест) число завдань кожного рівня підбирається таким, для яких загальне число операцій. З цією метою курс може бути розбитий на кілька розділів (модулів), представлених контрольними завданнями трьох рівнів складності і інформацією, необхідною для формування відповідних знань, умінь і навичок.

Завдання першого рівня складності (рівня знайомства з предметом) представляються завданнями теоретичного характеру (питаннями), ґрунтуються на виборчій методикою тестування та характерні діями по відтворенню інформації про об'єкт вивчення на рівні розуміння або пам'яті (число логічних операцій, необхідних для формування правильної відповіді, дорівнює двом).

Завдання другого рівня складності (рівня вмінь) представляються завданнями практичного характеру (питаннями) і характерні діями по використанню засвоєної інформації для вирішення найпростіших завдань на основі застосування готових способів вирішення (алгоритмів) без істотного їх перетворення. Число логічних і істотних операцій, необхідних для знаходження відповіді, дорівнює п'яти.

Завдання третього рівня складності (рівня «творчості») характерні діями по трансформації засвоєної інформації та представляються завданнями підвищеної складності, які вимагають від екзаменованих аналізу і синтезу раніше набутого досвіду з метою знаходження рішення в нових умовах і використання практично всього арсеналу знань, отриманих в процесі вивчення розділу або дисципліни. Тут діяльність екзаменованих пов'язана з необхідністю складання алгоритму розв'язання задачі шляхом перенесення раніше засвоєних умінь, а число логічних і істотних операцій, необхідних для виконання завдання, дорівнює дев'яти.

Таким чином, завдання кожного рівня складності якісно різні і для їх вирішення потрібні якісно різні рівні засвоєння, тобто діяльність на кожному рівні може бути здійснена лише за умови попереднього засвоєння відповідної інформації.У підсистемі планування та оцінювання знань будуть здійснені саме ці рівні складності, викладач має змогу самостійно обирати рівень складності тестових питань та загальний коефіцієнт складності тестування.

1.6 Система оцінювання тестування для підсистеми планування та оцінювання знань

Одним зі складних і суперечливих питань при проведенні тестування є проблема оцінювання знань. Найпоширенішим способом вирішення даної проблеми є використання дихотомічної системи оцінювання тестових завдань, в якій за кожну задачу можна отримати 0 або 1 бал. Дана система зручна при оцінюванні завдань з вибором однієї правильної відповіді з багатьох, тобто завдань закритого типу. У той же час існує певна різноманіття типів тестових завдань: закриті (багатоальтернативні і одноальтернативні), відкриті, на встановлення відповідності між елементами, на встановлення правильної послідовності, ситуаційні тестові завдання.

У попередньому розділі ми обрали три рівні складності для тестових питань, тепер слід обрати систему оцінювання тестування.

Загальна кількість операцій каскадного тесту у вигляді послідовності завдань різних рівнів складності визначається як умовами забезпечення його надійності, так і тривалістю іспиту.

Оптимальним (коефіцієнт надійності 0,75-0,8) можна вважати тест, що містить 70-80 операцій, що відповідає тривалості іспиту до трьох годин. При цьому кількість завдань кожного рівня може бути по-різному як в залежності від переслідуваної мети, так і в силу різного їх внеску у формування підсумкової оцінки, а при обробці результатів тестування враховується як кількість завдань певного рівня, так і число операцій, необхідних для їх вирішення.

2 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАВДАННЯ

2.1 Дослідження предмета, цілей і особливостей підсистеми планування та оцінювання знань

В рамках магістерської роботи необхідно розробити WEB-підсистему тестування студентів з наступними можливостями:

- наявність блоку реєстрації та авторизації. З повідомленням про проходження реєстрації на електронну пошту, шифруванням пароля за алгоритмом md5, а так само з перевіркою «анти-бот» і можливістю змінювати дані;
- добавити категорії на сторінки сайту за різними термінами. З можливістю для викладачів створити свої категорії сортування і включати в них терміни;
- створення тесту. З можливостями: сортування питань у випадковому порядку за бажанням викладача, відображення результатів і правильних відповідей, встановлення кількості спроб на проходження тесту, вибору налаштувань по проходженню тесту, настройки доступності тестів по даті і часу, установки порога успішної здачі тесту в процентах, виведення оцінки по завершення тестування (розрахунок оцінки відбувається в залежності від набраного відсотка);
- створення питання з вибором кількості правильних варіантів відповіді. З можливістю розміщення відповідей у випадковому порядку і з можливістю додавання даного питання в уже існуючий тест;
- створення питання з рукописним відповіддю. Викладач сам перевіряє відповідь студента;
- створення питання з можливістю встановити відповідність між пунктами підпитання, а також створення питання з можливістю встановити порядок підпитань;
- використання групової політики доступу для забезпечення безпеки. Студент, незареєстрований користувач, адміністратор і викладач;
- Призначення максимальної кількості балів за кожну повністю вірну відповідь.

2.2 Дослідження предметної області

Існує безліч різних систем інтернет тестування, що володіють як перевагами, так і недоліками. Основна проблема полягає в тому, що немає універсальної системи інтернет-тестування, позбавленої основних недоліків і яка володіє перевагами серед своїх аналогів, і якщо об'єднати переваги конкуруючих систем в один програмний продукт, то вийде дуже гнучка система, що дозволяє вирішувати дуже великий набір необхідних нам завдань. В ході розробки даного програмного засобу були виявлені наступні особливості системи інтернет тестування, які необхідно врахувати в даному продукті:

- реалізація можливості повернення до пропущених питань. Найчастіше користувачі пропускають складні питання і в першу чергу відповідають на прості, однак, повернутися до пропущених питань вже не можуть;
- відновлення сесії. Сесія – це механізм, що дозволяє однозначно ідентифікувати браузер і створює для цього браузера файл на сервері, в якому зберігаються змінні сеансу. Іноді користувачі випадково закривають сторінку на якій знаходяться або ж просто клацають мишею не на ті посилання. Для запобігання небажаних дій необхідно реалізувати відновлення сесії користувача. У сесії будуть зберігатися всі дані, які вводив користувач, де він був у момент закриття сторінки і куди натискав мишею. Це дозволить запобігти несанкціонованому припинення незакінченого тесту або ж втрату незбережених даних;
- реалізація масової реєстрації користувачів. Якщо тестування проходить одночасно велика група користувачів, то перед викладачем стане питання швидкості реєстрації учасників тестування і видачі їм унікальних логінів і паролів. Щоб оптимізувати цей процес, необхідно реалізувати масову реєстрацію в одну дію;
- реалізація гнучкої системи категорирования тестів. Коли користувач заходить на сайт інтернет тестування, то найчастіше не враховується при виборі цікавить його розділу. Категоріювання з дисциплін, спеціальностей і групам може значно скоротити час пошуку потрібного тесту;
- різним викладачам необхідні різні настройки для тестування, а значить необхідно реалізувати гнучку систему налаштування тестів

і питань в них. Наприклад, дозволити користувачам продовжити тестування, якщо вони його закрили, випадково або навмисно. Дозволити користувачам пропускати питання, якщо вони складні, і дозволити повертатися до пропущеним питань. Також заборонити переходити до наступного питання поки не буде дана правильна відповідь на поточне питання (для пробних тестів). Щоб виключити появу однотипних тестів необхідно реалізувати сортування питань у випадковому порядку або ж, навпаки, відключити сортування питань у випадковому порядку, якщо необхідний однаковий порядок питань в тесті. Для пробного тестування необхідно реалізувати можливість показувати правильні відповіді після проходження тесту або ж навпаки приховати їх. Є можливість позначення кількості спроб для проходження тесту. Якщо викладач захоче позначити час старту тестування і час його завершення, необхідно реалізувати настройки доступності тестування. Реалізовані процентні настройки тесту, дозволять встановити мінімальний поріг здачі тесту. А коли всі ці настройки реалізовані, необхідно реалізувати їх збереження, щоб знову не вибирати в нових тестах ті ж параметри;

- реалізація різних типів питань. Щоб найбільш повно розкрити знання користувача, а так само для того, щоб зробити тестування цікавим, необхідно додавати в тест питання різного типу. Наприклад, питання з одним або декількома правильними відповідями. Або ж питання з рукописним відповіддю, який перевіряється викладачем, після закінчення тестування. А так же, питання-відповідність, в якому необхідно вказати відповідність підпитань з термінами, і питання - порядок, де потрібно вибудувати терміни в правильному порядку;
- особливу увагу потрібно приділити можливості використання одного і того ж питання в різних тестах. Щоб викладач не витрачав час на створення аналогічно питання для іншого тесту, якщо теми для різних дисциплін перетинаються;
- реалізувати формування статистики на основі технології AJAX. Набагато зручніше і швидше робити вибірку за певним тестує введенням першої літери його імені або прізвища;
- для формування статистики, обліку всіх користувачів в системі та виділення їм певних прав доступу і привілеїв в системі необхідно реалізувати реєстрацію та авторизацію;

– у сучасних системах інтернет − тестування використовується механізм захисту від спам-програм −«капча». Зазвичай «капча» представлена у вигляді картинки з перекрученими символами. Розібрати, що зображено на малюнку складно, тому необхідно розробити більш просту систему захисту − математичну капчу. Відбувається генерація двох випадкових чисел, між якими ставиться знак складання. Користувачеві пропонується ввести результат складання в поле перевірки. Якщо результат збігається з даними зберігаються в БД, то користувач проходить перевірку, інакше, знову відбувається генерація «капчи»;
– для більш зручного моніторингу за результатами тестування по кожному користувачу необхідно реалізувати доступ до цих даних;
– реалізація групової політики доступу. Адміністратор, викладач, зареєстрований користувач, не зареєстрований користувач − всі ці користувачі системи мають свої правами доступу і своїми привілеями, необхідні для розмежування рівня доступу до даних.

Виключивши всі недоліки конкуруючих систем, і додавши всі їхні переваги, ми отримаємо сучасну і гнучку систему інтернет тестування.

2.3 Перелік термінів, визначень і скорочень

Повне найменування розроблюваної підсистеми «планування та оцінювання знань студентів», в подальшому іменується як «комплекс», «система». Список термінів, скорочень і визначень:

– БД – база даних;
– аккаунт − це обліковий запис, де зберігається персональна інформація користувача для входу у систему;
– облікова запис − запис, що містить відомості, які користувач повідомляє про себе деякою комп'ютерній системі, а так само відомості про права доступу і привілеї, виділених цього користувачеві;
– сесія − це механізм, що дозволяє однозначно ідентифікувати браузер і створює для цього браузера файл на сервері, в якому зберігаються змінні сеансу;
– словник − категорювання термінів;
– термін − категорирование тестів;
– категорювання − розбиття за категоріями;

- тест – це короткочасне, порівняно просто обставлене випробування, проведене в рівних для всіх випробовуваних умовах. За допомогою тестів проводиться тестування;
- тестування – це спеціально розроблена науково оптимізована атестаційна процедура, що дозволяє максимально об'єктивно оцінювати рівень досягнень людини і висловлювати ці можливості кількісно в формі чисел;
- спам – небажана реклама;
- реєстрація – ініціалізація користувача в системі, виділення йому прав доступу і привілеїв;
- незареєстрований користувач – людина, яка не пройшла реєстрацію;
- права доступу – сукупність правил, що регламентують порядок і умови доступу суб'єкта до інформації.
- викладач це є фізична особа, зареєстрована в системі з можливостями створювати і редагувати тести і питання в них, перевіряти відповіді студентів, створювати і видаляти словники і терміни в них;
- студент (від лат. Student – ретельно працюючий, займається) – учень вищого, у деяких країнах середнього, навчального закладу;
- питання з рукописним відповіддю – це такий тип питання, в якому студент може написати свою відповідь у вільній формі. Після чого ця відповідь буде перевірене викладачем;
- питання-порядок – такий тип питання, в якому студенту необхідно вибудувати підпитання (дії, дати та інше) в правильному порядку;
- питання-відповідність це такий тип питання, в якому студенту необхідно зіставити підпитання і правильні відповіді;
- електронна скринька користувача – адреса електронної пошти користувача;
- електронна пошта – технологія і надані нею послуги з пересилання і отримання електронних повідомлень (званих «листи» або «електронні листи») по комп'ютерній мережі.

3 ВИБІР І ОБГРУНТУВАННЯ АРХІТЕКТУРИ СИСТЕМИ ТА ПРОГРАМНИХ ЗАСОБІВ РЕАЛІЗАЦІЇ

Для реалізації інформаційної підсистеми оцінювання знань у віртуальному навчальному центрі, треба здійснити вибір і обґрунтування архітектуру системи. Згідно з визначеними функціями та бізнес-логікою яку повинна забезпечити система необхідно здійснити вибір сучасних програмних засобів реалізацій. Це є дуже важливим кроком при проектуванні системи, адже відкрита архітектура та потужні програмні середи розробки забезпечать сстемі змогу розширяти функціонал та легко маштабувати систему.

3.1 Вибір архітектури системи

Для інформаційної системи віртуального навчального центру було вибрано архітектуру «Триланковий клієнт-сервер» [3].

Архітектура була запропонована в 1995 році. Це подальше розширення триланкової архітектури, при якій функціональна частина колишнього «товстого» клієнта ділиться на дві частини:

- «тонкий», не інтелектуальний клієнт на робочій станції;
- середній рівень, керуючий всієї предметної логікою додатка.

На верхньому рівні залишається сервер бази даних. Переваги вибраної архітертури (рис.3.1):

- «тонкий» клієнт знижує вартість апаратного забезпечення робочої станції;
- централізація бізнес-логіки дозволяє централізувати супровід додатка (не потрібно інсталювати програмне забезпечення на всі робочі станції);
- модульність спрощує модифікацію і заміну;
- як наслідок рівномірного розподілу навантажень відбувається рвантаження сервера.

Триланкова архітектура природно відображається на середу WEB, де web-браузер виконує роль «тонкого» клієнта, а web-сервер – роль сервера додатків, при цьому триланкова архітектура може бути розширена до N-рівнів.

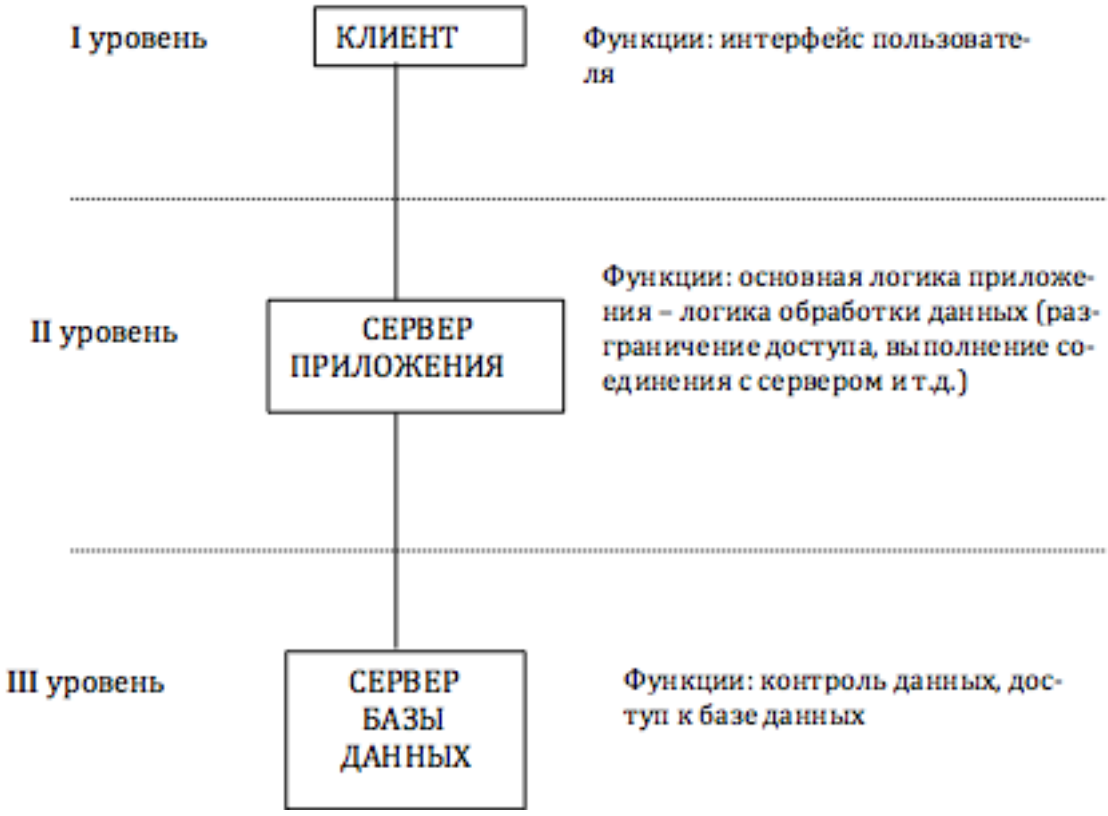


Рисунок 3.1 – Архітектура «Триланковий клієнт-сервер»

3.2 Вибір HTTP сервера

Для реалізації інформаційної системи потрібно обрати веб-сервер, що забезпечить обрану бізнес-логіку системи та буде здійснювати оброблення клієнтських запитів [4].

Веб-сервер – це сервер, що приймає HTTP-запити від клієнтів, зазвичай веб-браузерів, видає їм HTTP-відповіді, зазвичай разом з HTML-сторінкою, зображенням, файлом, медіа-потоком або іншими даними. Веб-сервер – основа всесвітньої павутини.

Веб-сервером називають як програмне забезпечення, що виконує функції веб-сервера, так і комп'ютер, на якому це програмне забезпечення працює.

Клієнти дістаються веб-сервера за URL-адресою потрібної їм веб-сторінки або іншого ресурсу.

Взаємодія між клієнтом і сервером здійснюється по протоколу HTTP-протокол передачі гіпертексту (рис. 3.2). Тобто в своїй основі протокол обміну між клієнтами та сервером Web є текстовим (незважаючи на те, що по ньому можлива передача і бінарних даних). Це робить його досить легким

для розуміння, програмної підтримки, налагодження програм, а також робить його зручним для міжплатформного взаємодії, тобто для спільної роботи клієнтів і серверів, реалізованих на різних платформах. Така взаємодія можна реалізувати і в бінарному протоколі, однак текстовий протокол більш прозорий. Протокол HTTP є одним з протоколів прикладного рівня в стеку протоколів TCP/IP і при цьому одним з найбільш затребуваних. Вже давно як весь стек протоколів TCP/IP, так і протокол HTTP використовується не тільки в мережі Internet, але і в локальних і корпоративних Internet мережах,здійснюючи взаємодія клієнтів і серверів в розподілених корпоративнихдодатках, які при необхідності легко переносяться і в Інтернет.В результаті огляду сучасних веб-серверів був вибраний сервер Apache в парі з веб-сервером Nginx.

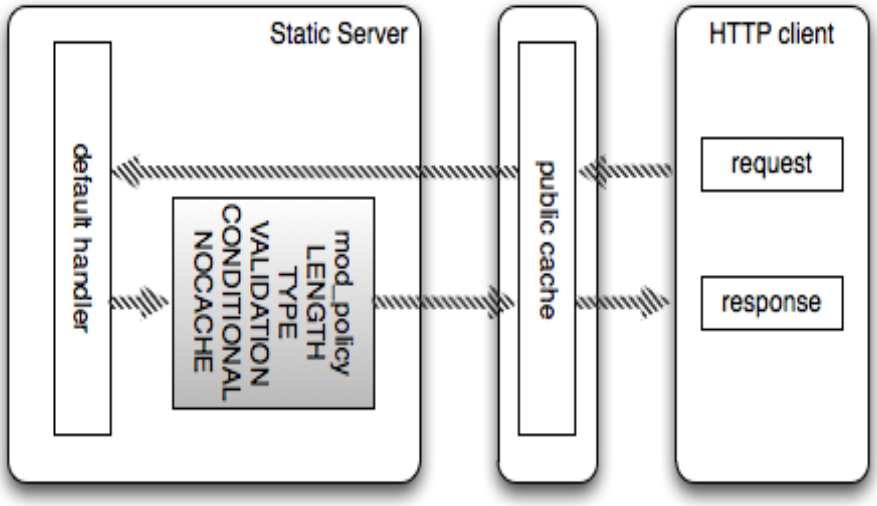


Рисунок 3.2 – Взаємодія між клієнтом і веб-сервером по протоколу HTTP

3.2.1Сервер Apache

Для реалізації інформаційної системи потрібен сервер для обробки запитів та повернення результатів скрипнів на мові PHP.

Web-сервер Apache є самостійним, некомерційним, вільно розповсюджуваним продуктом. Продукт підтримує безліч можливостей, багато з яких реалізовані як скомпільовані модулі, які розширюють основні функціональні можливості. Вони різняться від серверної підтримки мов програмування до схем аутентифікації. Існують інтерфейси для підтримки мов програмування Perl, Python, Tcl і PHP.

Apache передусім використовується даля передачі через HTTP статичних та динамічних веб-сторінок у всесвітній павутині. Багато веб-застосунків спроектовано, зважаючи на середовище і можливості, які надає цей веб-сервер.

Має власну мову конфігураційних файлів, заснований на блоках директив. Практично всі параметри ядра можуть бути змінені через конфігураційні файли. Більша частина модулів має власні параметри.

Одна із особливостей являється файл .htaccess. Не потрібно настроювати веб-додаток у глобальному файлі конфігурації сервера, а достатньо лише створити локальний файл .htaccess(файл можно сворювати в кожній дректорії додатку).

Apache конфігурується зміною службових файлів в каталозі /etc/httpd/conf/. Головний конфігураційний файл веб-сервера – httpd.conf. Конфігураційні директиви можуть розміщуватися в різних файлах, які включають в основний конструкцією Include ім'я_файла.conf.

Спеціальний модуль системного рівня Multi-Processing Module (MPM) дає можливість оптимізувати Apache в умовах конкретної операційної системи, надаючи ще один варіант доступу до системних сервісів. Після стартової початкової ініціалізації ядро передає управління модулю MPM, який підтримує пул робочих процесів / потоків, реалізує інтерфейс між сервером і даною операційною системою, оптимізуючи роботу сервера.

MPM має 2 основних режими роботи:

– Prefork –цетрадиційний non-threaded варіант;
– Worker – багатопоточний варіант, якому властивий менший витрата пам'яті.

Більшість модулів Apache мають безпосереднє відношення до обробки клієнтських запитів, беручи участь у цьому процесі в певній послідовності. Такий підхід дозволяє кожному модулю сфокусуватися на вузькому аспекті обробки.

Веб-сервер повертає клієнту відповідь, що формується генератором контенту. Будь модуль може зареєструвати свій генератор за допомогою директив SetHandler або AddHandler у файлі httpd.conf. Якщо такої реєстрації немає, то генератор за замовчуванням просто повертає файл, одержуваний безпосередньо із запиту. Модулі, які реалізують свої власні генератори, називаються контент-генераторами. Такий генератор в принципі може управляти всіма функціями. Наприклад, CGI програма, що одержала запит, генерує відповідь, повністю контролюючи цей процес.

3.2.2Сервер Nginx

Для ефективної роботи інформаційної системи потрібен сервер для балансування навантаження на головний веб-сервер Apache, який займається виключно виконанням бізнес-логіки.

Конфігурація HTTP-сервера 'nginx' розділяється на віртуальні сервери (директива server). Віртуальні сервери поділяються на локації (location). Для віртуального сервера можливо задати адреси і порти, на яких будуть прийматися з'єднання, а також імена, які можуть включати * для позначення довільній послідовності в перший і останній частині, або задаватися регулярним виразом. Локації можуть задаватися точним URI, частиною URI, або регулярним виразом. location'и можуть бути налаштовані для обслуговування запитів зі статичного файлу, проксування на http, fastcgi чи memcached сервер.

Nginx містить модуль географічної класифікації клієнтів за IP-адресою. У його основу входить база даних відповідності IP-адрес географічному регіону, представлена у вигляді Radix tree (стиснуте префіксне дерево або стиснений бор) в оперативній пам'яті. nginx попередньо розподіляє перші кілька рівнів дерева, таким чином, щоб вони займали рівно 1 сторінку пам'яті. Це гарантує, що при пошуку IP-адреси для перших декількох вузлів при трансляції адреси завжди знайдеться запис в TLB.

Основні функції Npinx:

- обслуговування статичних запитів, індексних файлів, автоматичне створення списку файлів, кеш дескрипторів відкритих файлів;
- акселероване проксіювання з підтримкою кешування;
- акселерована підтримка FastCGI і memcached серверів, простий розподіл навантаження і відмовостійкість;
- модульність, фільтри, gzip, byte-ranges (докачка), chunked відповіді, HTTP-аутентифікація, SSI-фільтр;
- вкладені запити на одній сторінці виконуються паралельно;
- підтримка SSL;
- експериментальна підтримка вбудованого Perl.

Для даного застосування адміністратора підходили два лідери серед HTTP-серверів:

- Apache – відкритий веб-сервер Інтернет для UNIX-подібних, Microsoft Windows, Novell NetWare та інших операційних систем.

– Nginx – вільний веб-сервер і проксі-сервер. Є версії для сімейства Unix-подібних операційних систем (FreeBSD, GNU/Linux, Solaris, Mac OS X) та Microsoft Windows.

3.3 Опис схеми співпраці двох HTTP-серверів

Для роботи веб-додатку використовуються 2 веб-сервери одночасно. Nginx використовується як проксі-сервер, що дозволяє знизити навантаження на Apache (рис. 3.3).

Nginx прослуховує http (80) порт, якщо http запит запрошує медіа-контент, css стилі, js файли Nginx опрацьовує запит сам, всі інші записи проксуються на порт який прослуховує Apache. Apache приймає запити і опрацьовує уже як повноцінний HTTP-сервер, наприклад, запускає php-файл і педає необхідні дані для опрацювання.

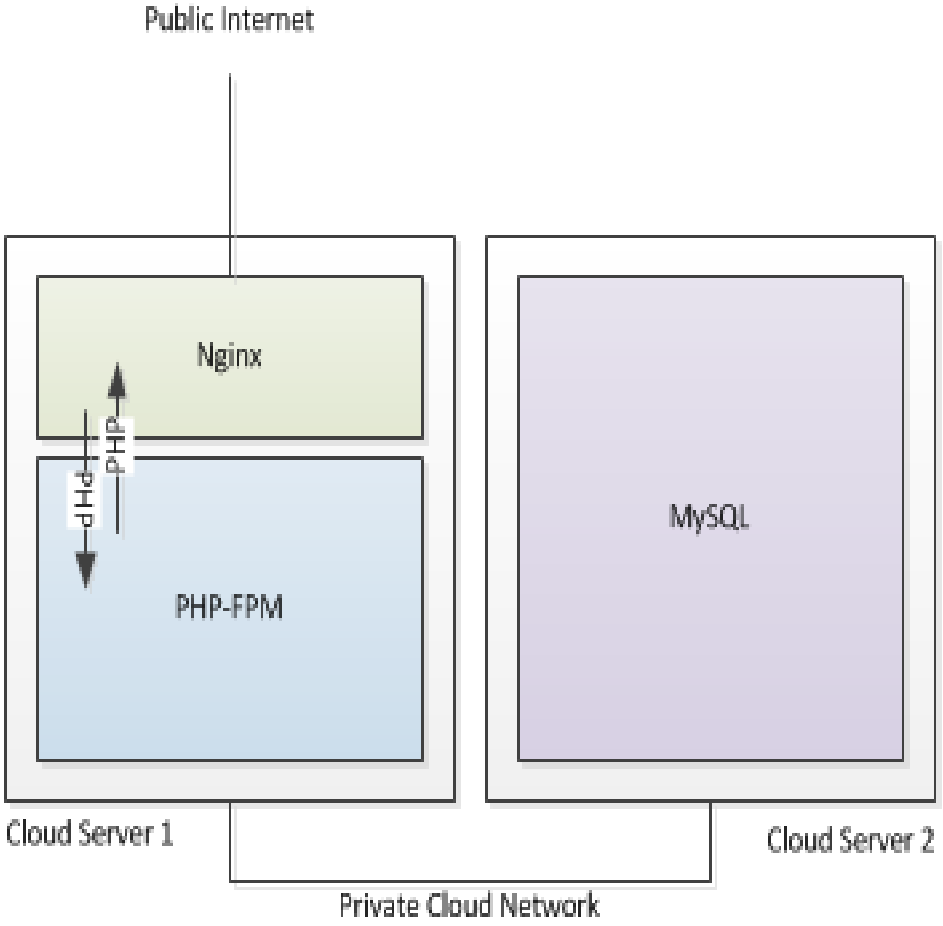


Рисунок 3.3 – Схема співпраці двох HTTP-серверів

Дана схема дозволяє знизити навантаження на Apache, що позитивно впливає на час обробки запиту. Nginx в свою чергу, як більше легший HTTP-сервер опрацьовує запити, які не мають стосунку до бізнес-логіки додатку.

3.4 Вибір системи управління базами даних

Для зберігання даних інформаційної системи віртуального навчального центру потрібна база даних, вимогою якої є висока стабільність роботи та швидкість виконання запитів.

Реляційні СУБД є найбільш поширеним видом систем управління базами даних на даний момент. Вони дійсно легкі у використанні. Реляційні СУБД мають таку назву, тому вони зберігають впорядковані дані в таблицях. Реляційна модель ґрунтується на зберіганні впорядкованих даних у стовпцях і рядках. Дані можуть бути пов'язані в межах однієї таблиці або різних таблиць. Типи реляційних СУБД можуть поступатися в продуктивності в порівнянні з іншими видами систем управління базами даних, проте вони не мають проблем з обчислювальною потужністю і пошуком пам'яті в сучасних ПК. Дані в цьому типі СУБД зберігаються в точно визначеному вигляді, а управління ними, зазвичай, виконується за допомогою мови програмування SQL (Structured Query Language). Так само можна вказати найбільш популярні типи СУБД – Oracle, MSSqlServer, PostgreSQL і багато інших [5].

Oracle використовується для GRID вичислень, в данній інформаційній системі використовується лише один сервер для зберігання данних.

MS SQL Server – для данного СУБД потрібно використовувати операційну систему Microsoft Windows Server, але для данної системи вкористовується операційна система Linux, що не дозволяє використати данну субд.

PostgreSQL – молода СУБД, яка являється прям конкурентом MySql, але з врізаним функціоналом.

Для розробки інформаційно-пошукової використовувати СУБД середнього масштабу і продуктивності. На сьогоднішній день СУБД MySQL є однією з найвідоміших, надійних і швидких з усього сімейства існуючих СУБД.

MySQL це вільна система керування реляційними базами даних.

MySQL є рішенням для малих і середніх додатків. Входить до складу серверів WAMP, LAMP і в портативні збірки серверів Денвер, XAMPP. Зазвичай MySQL використовується як сервер, до якого звертаються локальні або видалені клієнти, проте в дистрибутив входить бібліотека внутрішнього сервера, що дозволяє включати MySQL в автономні програми [6].

MySQL написаний під десятки видів операційних систем. ЦеiFreeBSD, OpenBSD, MacOS, OS/2, SunOS, Win9x/00/NTiLinux. Сьогодні MySQL особливопоширенанаплатформах Linux і Windows. Причомунаостаннійзустрічаєтьсянабагаторідше.

MySQL, якібудь-якаіншаСУБДявляєсобоюпрограму-сервер, яказнаходитьсявпам'ятікомп'ютераіобслуговує TCP порт. У випадку з MySQL, номером порту буде число 3306. А клієнтська програма, будь то CGI-додаток на Perl або програмний продукт на C, з'єднується з СУБД з цього порту і посилає йому рядки на SQL. Той у свою чергу їх інтерпре-тує, виконуючи необхідні дії, і відсилає результати запиту назад клієнтові. Таким способом відбувається спілкування сервера баз даних з клієнтськи-ми датками [7].

База даних, яку сервер MySQL використовує для зберігання внутрішньою інформації про користувачів, за замовчуванням має ім'я mysql. У цій базі даних певні таблиці для зберігання інформації користувача облікових записів.

Гнучкість СУБД MySQL забезпечується підтримкою великої кількості типів таблиць: користувачі можуть вибрати як таблиці типу MyISAM, що підтримують повнотекстовий пошук, так і таблиці InnoDB, що підтримують транзакції на рівні окремих записів. Більш того, СУБД MySQL поставляється із спеціальним типом таблиць EXAMPLE, що демонструє принципи створення нових типів таблиць. Завдяки відкритій архітектурі і GPL-ліцензуванню, в СУБД MySQL постійно з'являються нові типи таблиць.

MySQL має API для мов Delphi, C, C ++, Java, Лисп, Perl, PHP, Python, Ruby, Smalltalk, Компонентний Паскаль і Tcl, бібліотеки для мов платформи .NET, а також забезпечує підтримку для ODBC за допомогою ODBC-драйвера MyODBC [8].

MyODBC являє собою драйвер ODBC рівня 0 (з деякими можливостями рівнів 1 і 2) для приєднання сумісного з ODBC додатки до MySQL. MyODBC працює на всіх системах Microsoft Windows і на більшості платформ Unix.

В результаті проведеного аналізу сучасних СУБД була обрана система управління базами даних MySQL можливості якої забезпечити коректне функіонування системи віртуального навчального центру та коректну роботу даних.

3.5 Вибір мови програмування

Для інформаційної системи віртуального навчального центру потрібно обрати мову програмування за допомогою якої буде здійснена розробка скриптів, які реалізують визначені функції системи [9]. На вибір мови програмування впливають такі фактори як:

– характер розв’язуваної задачі;
– наявні системні бібліотеки;
– підтримувані компілятором платформи.

За характером розв'язуваної задачі, для програмування інтерпретатора потрібно мова програмування, що дозволяє:

– гнучко працювати з динамічно виділяється пам'яттю;
– мати об'єктно-орієнтоване розширення;
– мати засоби обробки виняткових ситуацій;
– отримувати високошвидкісний код.

Зазначеним вимогам задовольняє мова PHP.

PHP – скриптова мова програмування загального призначення, інтенсивно застосовується для розробки веб-додатків. В даний час підтримується переважною більшістю хостинг-провайдерів і є одним з лідерів серед мов програмування, що застосовуються для створення динамічних веб-сайтів [10].

Додатки на PHP швидко інтегруються із веб-сервером Apache. Після установки веб-сервера не потрібно добавляти бібліотеки для запуску php файлів. PHP був вибраний як мова програмування, яка розроблена для розробки веб-додатків, а не мова загального призначення, наприклад, як Java, Ruby, Python.

Застосовується в області веб-програмування, зокрема серверна частина, PHP – один з популярних сценарних мов (разом з JSP, Perl і мовами, використовуваними в ASP.NET) завдяки своїй простоті, швидкості виконання, багатій функціональності, багатоплатформеності та поширенню вихідних кодів на основі ліцензії PHP.

PHP є мовою програмування з динамічною типізацією, що не вимагає вказівки типу при оголошенні змінних, так само як і самого оголошення змінних. Перетворення між скалярними типами найчастіше здійснюються неявно без додаткових зусиль (втім, PHP надає широкі можливості і для явного перетворення типів).

Порядок елементів і їх ключів зберігається. Не зовсім коректно називати php-масиви масивами, насправді це, швидше за все, упорядкований хеш. Можливо несподіване поведінку при використанні циклу for з лічильником

замість foreach. Так, наприклад, при сортуванні масиву з чисельними індексами функціями із стандартної бібліотеки, упорядковано і ключі теж.

Інтерпретатор складається з ядра і модулів, «розширень», що представляють собою динамічні бібліотеки. Розширення дозволяють доповнити базові можливості мови, надаючи можливості для роботи з базами даних, сокетами, динамічною графікою, криптографічними бібліотеками, документами формату PDF і тому подібним. Будь-який бажаючий може розробити своє власне розширення і підключити його. Існує величезна кількість розширень, як стандартних, так і створених сторонніми компаніями і ентузіастами, проте в стандартне постачання входить лише декілька десятків добре зарекомендували себе. Безліч розширень доступно в репозиторії PECL.

Інтерпретатор PHP має спеціальний конфігураційний файл – php.ini, що містить безліч налаштувань, зміна яких впливає на поведінку інтерпретатора. Є можливість відключити використання ряду функцій, змінити обмеження на використовувану скриптом оперативну пам'ять, час виконання, обсяг завантажуваних файлів, налаштувати журналирование помилок , роботу з сесіями та поштовими сервісами, підключити додаткові розширення, а також багато іншого. Можливо дроблення великого конфігураційного файлу на частини. Наприклад, широко поширена практика винесення налаштувань розширень в окремі файли. Параметри інтерпретатора можуть бути перевизначені в файлах конфігурації HTTP-сервера (наприклад, .htaccess в Apache) або в самому скрипті під час виконання за допомогою команди ini_set.

Для швидкої розробки додатків на PHP було створено безліч фреймворків, найбільш популярними з яких є Zend Framework, CakePHP, Symfony, CodeIgniter, Kohana і Yii. Основні переваги такої розробки – це надання можливості будувати проект за допомогою патерну MVC [11].

Основні переваги PHP: постійно вдосконалюється; працює на UNIX та Windows платформах; допускає роботу з більшістю СУБД; має широ-кий набір функцій (більше 3 тис.); допускає об'єктно-орієнтоване програмування; здатний використовувати протоколи HTTP, FTP, SNMP, NNTP, POPS [12]. Дозволяє виконувати всі операції, що і перераховані його конкуре-нти, і навіть працювати з файлами графіки. Можна також запускати PHP-скрипти як інтерпретуються файли і компілювати виконувані додатки (у тому числі з підтримкою графічного інтерфейсу GTK).

HTML стандартна мова розмітки веб-сторінок в Інтернеті. Більшість веб-сторінок створюються за допомогою мови HTML (або XHTML). Доку-

мент HTML оброблюється браузером та відтворюється на екрані у звичному для людини вигляді

HTML є похідною мовою від SGML, успадкувавши від неї визначення типу документу та ідеологію структурної розмітки тексту. HTML разом із каскадними таблицями стилів та вбудованими скриптами – це три основні технології побудови веб-сторінок [13].

Елементи являють собою базові компоненти розмітки HTML. Кожен елемент має дві основні властивості: атрибути та зміст (контент). Існують певні настанови щодо кожного атрибута та контента елемента, які треба вонувати задля того, щоб HTML-документ був визнаний валідним [14].

Більшість з атрибутів елементу являє собою пару «назва-значення», розділених між собою знаком рівняння, та записаних у початковому тегу оразу після назви елемента. Значення атрибуту може бути окреслено лапками (подвійними або одиничними), також, якщо значення атрибуту складається з певних символів, його можна не виділяти лапками зліва. Проте, не виділення значення атрибутів в лапки вважається небезпечним кодом. На відміну від атрибутів виду «назва-значення», є певні атрибути, що впливають на еемент, назва яких лише з'явилась в початковому тегу (наприклад, атрибут ismap елементу img).

Для перегляду HTML-розмітки документа можна використовувати будь-який текстовий редактор. Для перегляду документу, відтвореного за правилами HTML-розмітки, використовується браузер.

Всесвітня павутина складається в основному з HTML документів, переданих з веб-серверів для браузерів, використовуючи протокол HTTP. До того ж, HTTP використовується для передачі зображень, звуків, відео та іншого супутнього контенту. Для правильного відтворення документу браузером, окрім нього самого передається ще й інша інформація (метадані), у якій зазвичай міститься визначення MIME типу (наприклад, text/html або application/xhtml+xml) та кодової таблиці документа.

Каскадні таблиці стилів – спеціальна мова, що використовується для опису сторінок, написаних мовами розмітки даних.

Найчастіше CSS використовують для візуальної презентації сторінок, написаних HTML та XHTML, але формат CSS може застосовуватися до інших видів XML-документів.

Специфікації CSS були створені та розвиваються Консорціумом Всесвітньої мережі.

CSS має різні рівні та профілі. Наступний рівень CSS створюється на основі попередніх, додаючи нову функціональність або розширюючи вже наявні функції. Рівні позначаються як CSS1, CSS2 та CSS3. Профілі це сукупність правил CSS одного або більше рівнів, створені для окремих типів пристроїв або інтерфейсів. Наприклад, існують профілі CSS для принтерів, мобільних пристроїв тощо.

CSS (каскадна або блочна верстка) прийшла на заміну табличній верстці веб-сторінок. Головна перевага блочної верстки це розділення змісту сторінки (даних) та їхньої візуальної презентації.

JavaScript (JS) – динамічна, об'єктно-орієнтована мова програмування. Реалізація стандарту ECMAScript. Найчастіше використовується як частина браузера, що надає можливість коду на стороні клієнта (такому, що виконується на пристрої кінцевого користувача) взаємодіяти з користувачем, керувати браузером, асинхронно обмінюватися даними з сервером, змінювати структуру та зовнішній вигляд веб-сторінки. Мова JavaScript також використовується для програмування на стороні серверу (подібно до таких мов програмування, як Java і C#), розробки ігор, стаціонарних та мобільних додатків, сценаріїв в прикладному ПЗ (наприклад, в програмах зі складу Adobe Creative Suite), всередині PDF-документів тощо.

JavaScript класифікують як прототипну (підмножина об'єктно-орієнтованої), скриптову мову програмування з динамічною типізацією. Окрім прототипної, JavaScript також частково підтримує інші парадигми програмування (імперативну та частково функціональну) і деякі відповідні архітектурні властивості, зокрема: динамічна та слабка типізація, автоматичне керування пам'яттю, прототипне наслідування, функції як об'єкти першого класу.

jQuery – популярна JavaScript-бібліотека з відкритим сирцевим кодом. Згідно з дослідженнями організації W3Techs, JQuery використовується понад половиною від мільйона найвідвідуваніших сайтів. jQuery є найпопулярнішою бібліотекою JavaScript, яка посилено використовується на сьогоднішній день.

Синтаксис jQuery розроблений, щоб зробити орієнтування у навігації зручнішим завдяки вибору елементів DOM, створенню анімації, обробки подій, і розробки AJAX-застосунків. jQuery також надає можливості для розробників, для створення плагінів у верхній частині бібліотеки JavaScript. Використовуючи ці об'єкти, розробники можуть створювати абстракції для

низькорівневої взаємодії та створювати анімацію для ефектів високого рівня. Це сприяє створенню потужних і динамічних веб-сторінок.

3.6 Вибір PHP Framework

Для реалізації інформаційної системи потрібен фреймворк, який надасть смогу реалізувати логічну структуру системі та забезпечить розробника системи більшістю необхідних бібліотек.

Програмний фреймворк (англ. software framework) – це готовий до використання комплекс програмних рішень, включаючи дизайн, логіку та бзову функціональність системи або підсистеми. Відповідно – програмний фреймворк може містити в собі також допоміжні програми, деякі бібліотеки коду, скрипти та загалом все, що полегшує створення та поєднання різних конентів великого програмного забезпечення чи швидке створення готового і не обов'язково об'ємного програмного продукту. Побудова кінцевого подукту відбувається, зазвичай, на базі єдиного API.

CodeIgniter – популярний MVC фреймворк з відкритим вихідним кодом, написаний на мові програмування PHP, для розробки повноцінних веб-систем і додатків. Розроблений компанією EllisLab.

Особливості CodeIgniter – відрізняє простота, яка досягається завдяки наступним факторам:

- якісна і повна документація з прикладами, а також велика спільнота і Wiki;
- не використовуються генератори коду з командного рядка;
- codeIgniter працює практично на будь-якому хостинговому плані, який має підтримку PHP версії 5.1 і вище;
- codeIgniter вважається одним з найшвидших і не вимогливих до ресурсів фреймворків;
- малий розмір дистрибутива (розмір версії 2.1.1 складає всього 2.2 Мб);
- фреймворк дозволяє відмовитися від моделей.

Документація CodeIgniter пропагує «товсті» контролери і «тонкі» моделі. Валідація і побудова бізнес-логіки відбуваються в основному в кнтролері. Незважаючи на це, фреймворк дає свободу розробнику, тому він може самостійно вибрати підхід до розробки додатка.

Можливості CodeInteger:

- підтримка баз даних MySQL, PostgreSQL, MSSQL, SQLite, Oracle;
- підтримка псевдо-ActiveRecord, який здебільшого повторює синтаксис мови SQL;
- легко розширюється система за рахунок можливості використання сторонніх і самописних бібліотек, а також доповнення або превизначення існуючих;
- фреймворк містить в собі безліч необхідних бібліотек, які сворюють функціонал для роботи з файлами, відправки електронних листів, валідації форм, підтримки сесій, роботи з зображеннями і так далі;
- володіє можливістю кешування на стороні сервера SQL-запитів і генеруються html-сторінок. З версії 2.0 для кешування можуть використовуватися XCache або APC;
- у 2011 році створили доповнення, яке робить можливим підтримку міграцій;
- підтримка модульності (HMVC) за допомогою доповнень;
- незважаючи на відсутність ORM в стандартом пакеті, існує можливість використання PHP ActiveRecord, Doctrine, Propel та деяких інших ORM після невеликих змін або доповнень у вихідному коді фреймворку.

Для реалізації системи віртуального навчального центру був обраний програмний фреймворк з відкритим кодом, який забезпечив необхідними інструментами які знаходяться в його бібліотеках для реалізації інтерфейсу адміністратора.

3.7 Система управління версіями GIT

Для управління версіями та кодом при розробки інформаційної системи віртуального навчального центру потрібна система управління версіями. Також за допомогою системи управління версіями буде відбуватися резервування клієнтських файлів.

Git – розподілена система керування версіями файлів та спільної роботи. Git є однією з найефективніших, надійних і високопродуктивних систем керування версіями, що надає гнучкі засоби нелінійної розробки, що базуються на відгалуженні і злитті гілок. Для забезпечення цілісності історії та стійкості до змін заднім числом використовуються криптографічні методи, також можлива прив'язка цифрових підписів розробників до тегів і комітів.

Віддалений доступ до репозиторіїв Git забезпечується git-демоном, SSH або HTTP сервером. TCP-сервіс git-daemon входить у дистрибутив Git і є разом з SSH найпоширенішим і надійним методом доступу. Метод доступу HTTP, незважаючи на ряд обмежень, дуже популярний в контрольованих мережах, тому що дозволяє використання існуючих конфігурацій мережевих фільтрів. Дана система дозволяє безперебійно перенести змінений код PHP на головний сервер для подальшої обробки клієнтських запитів.

Git використовується для зберігання історії змін коду, що дозволяє аналізувати роботу коду після його змін, також Git полегшує розробку ПО декільком розробникам.

За допомогою Git новий код поміщається на головний сервер з комп'ютера розробника. За допомогою Git будуть розроблені Bash-скрипти для резервування завантажених файлів клієнтів.

4 ПРОЕКТУВАННЯ ІНФОРМАЦІЙНОЇ ПІДСИСТЕМИ

4.1 Проектування інформаційної підсистеми оцінювання знань за допомогою методології функціонального моделювання SADT

При проектуванні віртуальної інформаційної підсистеми оцінювання знань була обрана методологія функціонального моделювання SADT (Structured Analysis and Design Technique) – стандарт IDEF0.

Методологія SADT – одна з найвідоміших методологій аналізу та проектування систем. Вона є, мабуть, єдиною методологій, що відбиває такі храктеристики, як управління, зворотній зв'язок і ресурси. Інша особливість SADT полягає в тому, що вона розвивалася як мова опису функціонування систем загального вигляду, тоді як в інших структурних методологіях упор частіше робиться на проектування програмного забезпечення.

IDEF0 – методологія та стандарт функціонального моделювання. За допомогою графічної мови IDEF0, інформаційно-пошукова система фармацевтичних фірм постає у вигляді набору взаємопов'язаних функціональних блоків. Моделювання засобами IDEF0, як правило, є першим етапом вивчення системи.

Контекстна діаграма є вершиною деревовидної структури діаграм і являє собою саме загальний опис системи та її взаємодія з зовнішнім середовищем. Проектування починається з представлення системи як єдиного цілого – одного функціонального блоку з граничними стрілками, які простираються за межі аналізованої області.

Головна робота підсистеми оцінювання знань студентів відображена на контекстній діаграмі яка не сильно відрізняється від контекстної діаграми основної системи віртуального центру.На вхід подається інформація про клієнта, навчальні матеріали та завдання клієнта. Головна робота керується: особливостями хостингу, правилами публікації матеріалів, правилами публікації додатка, платіжним сервісом та правилом публікації тестових питань. Виходом є: клієнт з готовим завданням, клієнт з навчальним матеріалом, клієнтська монетізація та успішно пройдене тестування.

Після опису системи в цілому проводиться розбиття її на великі фрагменти. Цей процес називається функціональна декомпозиція, а діаграми, які

описують кожен фрагмент і взаємодію фрагментів, називаються діаграмами декомпозиції.

Після декомпозиції контекстної діаграми проводиться декомпозиція кожного великого фрагмента системи на більш дрібні і так далі, до досягнення потрібного рівня подробиці опису. Після кожного сеансу декомпозиції проводяться сеанси експертизи – експерти предметної області вказують на відповідність реальних процесів створеним діаграмам. Знайдені невідповідності виправляються, і тільки після проходження експертизи без зауважень можна приступати до наступного сеансу декомпозиції. Так досягається відповідність моделі реальним процесам на кожному рівні декомпозиції моделі. Синтаксис опису системи в цілому і кожного її фрагмента однаковий у всій моделі. Контекстна діаграма інформаційної системи віртуального навчального центру наведена на рис. 4.1.

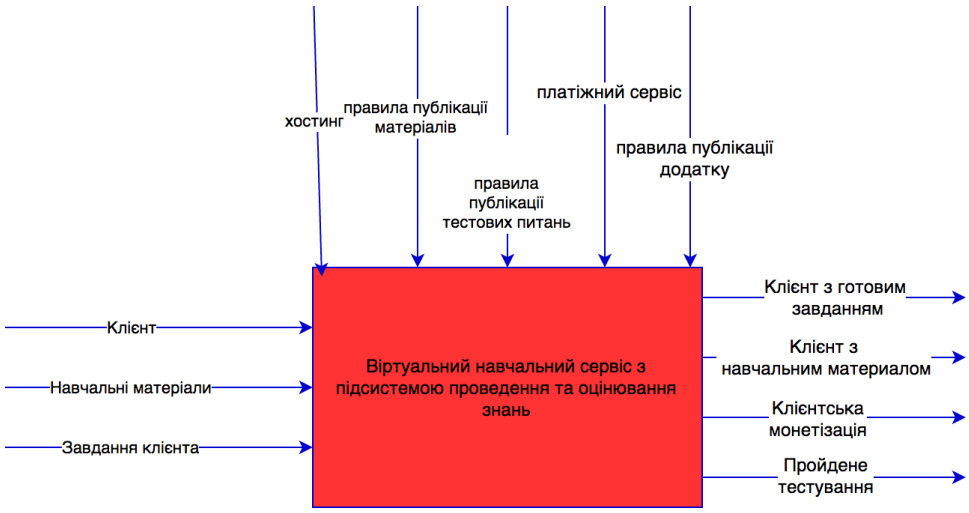


Рисунок 4.1 – Контекстна діаграма інформаційної системи віртуального навчального центру з підсистемою оцінювання знань

Після декомпозиції контекстної діаграми отримуємо 6 блоків. Ці блоки представляють основні під функції початкової функції.

Функція «Розробка web-системи» включає в себе повну розробку інформаційної системи на локальному комп’ютері. Включає в себе розробку інтерфейсу, скриптів, розробку та наповнення баз даних. Входом для цієї функції є навчальні матеріали, та як вони потрібні для заповнення БД. Функція управляється за допомогою правил публікації матеріалів та платіжним сервісом. Механізмом функції є програмне забезпечення, яке потрібне для розробки. І нарешті, результатом роботи є готова Web-система.

Функція «Розробка підсистеми» включає в себе розробку підсистеми для проходження тестування отриманних знань при використанні основної системи віртуального навчального центру. Входом для роботи служать тестові питання написанні викладачем. Виходом є розроблена підсистема.

Функція «Розміщення Web-системи» служить для можливості отримати доменне імя, та після цього змогу розмістити систему на хостінг, також результатом роботи є розроблений Rest API. Входом для роботи є готова Web-система для розміщення. Механізмом є програмне забезпечення.

Функція «Розробка додатка» призначена для того, щоб надати змогу реалізувати мобільний додаток під різні платформи. Входом для роботи є рзроблений Rest API. Управляється робота правилами публікації матеріалів. Механізмом вважається програмне забезпечення. Результатом є готовий додаток.

Функція «Публікація додатка» дає змогу публікувати додаток, який є допоміжною ланкою системи. Входом для роботи є готовий додаток для розміщення. Управляється робота правилами публікації додатка. Механізмом є програмне забезпечення. Результатом є опублікований додаток.

Функція «Реалізація» призначена для того, щоб розробити все те, що потрібно для правильної роботи інформаційної системи навчального центру. Наприклад, це може бути: реєстрація, каталог матеріалів, корзина і т.д.

У цієї роботи є чотири входи: доменне ім’я, готовий додаток, клієнт та завдання клієнта. Управляється правилами публікації матеріалів. Механізмом є – програмне забезпечення. Виходом даної роботи є: клієнт з готовим завданням, клієнт з навчальним матеріалом, клієнтська монетизація.

Діаграма декомпозиції наведена на рис. 4.2.

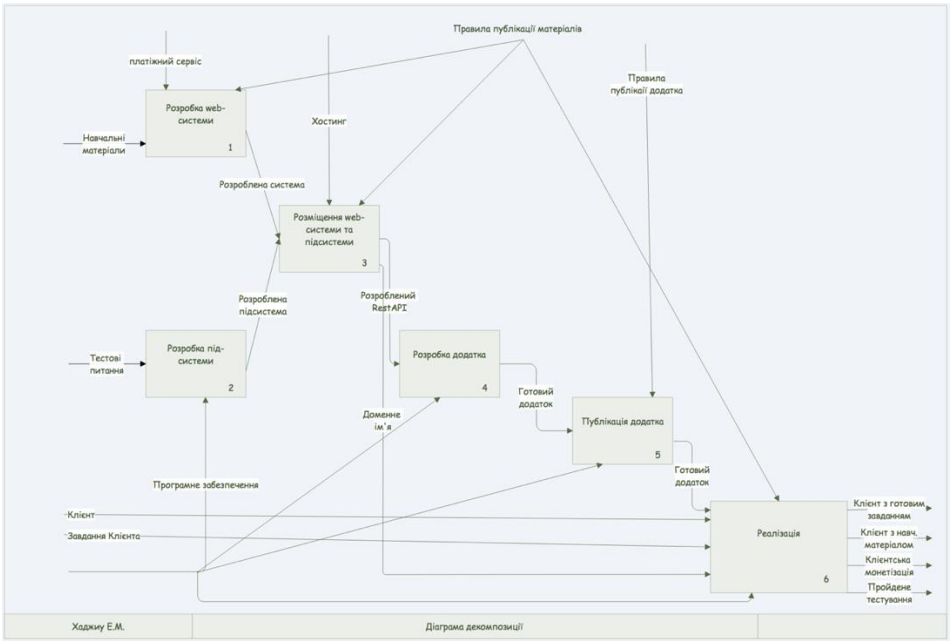


Рисунок 4.2 – Діаграма декомпозиції інформаційної системи віртуального навчального центру з підсистемою оцінювання знань

При декомпозиції першого А1 виділяються наступні блоки:

«Створення інтерфейсу» – входу у даної роботи немає, управління – правила публікації матеріалів та платіжний сервіс; механізм – програмне збезпечення; вихід – інтерфейс Web-системи.

Для адміністратора потрібно розробити інтерфейс: для управління користувачами, для модерації курсів та питань користувачів, для модерації відгуків та рейтингу навчальних матеріалів, для обробки запитів виводу коштів користувача, для зв’язку з клієнтами.

«Створення БД» – даний блок управляється правилами розробки БД; робота виконується за допомогою програмного забезпечення; виходом є – готова БД.

«Розробка скриптів» – входом є готова БД – вихід попередньої роботи; управлінням є, як і в попередньому блоці, правила розробки скриптів, але додалися правила публікації матеріалів; механізм, як і у всіх роботах, є – програмне забезпечення; вихід – робоча Web-система.

«Наповнення БД» – робота має два входи: готова БД, робоча Web-система, навчальні матеріали; керується правилами публікації матеріалів; мханізмом є програмне забезпечення. Вихід з даної роботи є готова рроблена Web-система. Діаграма декомпозиції першого А1 («Розробка web-системи») блоку представлена на рис. 4.3.

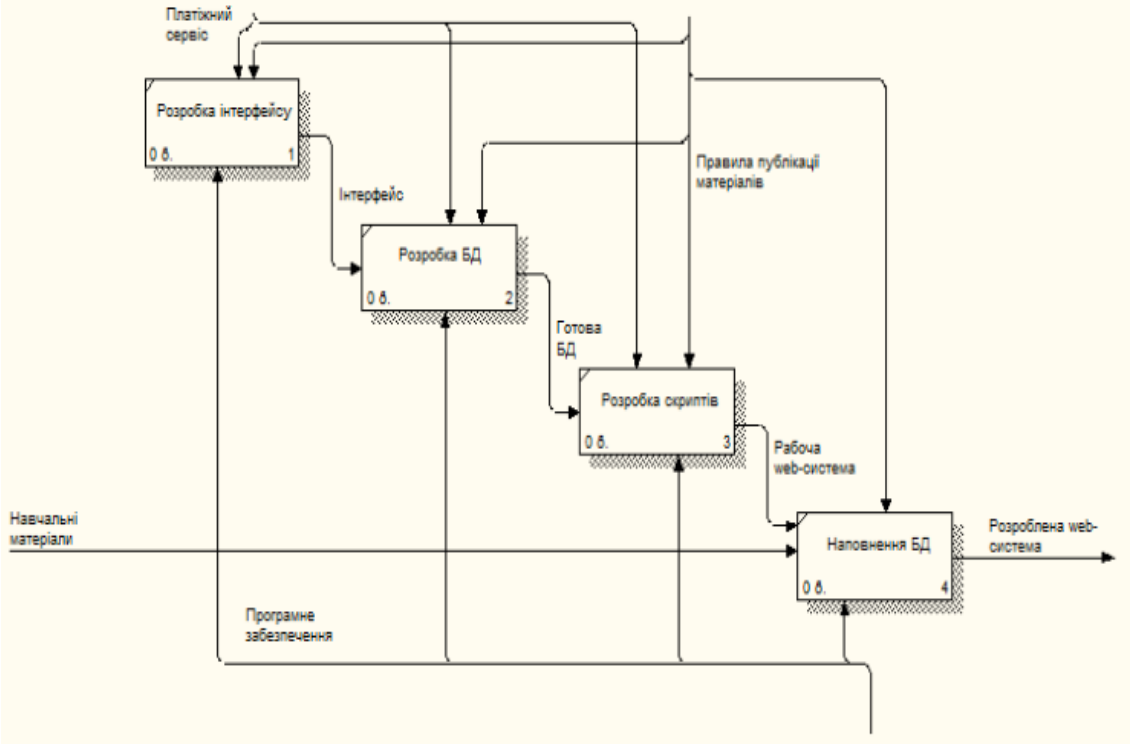


Рисунок 4.3 – Діаграми декомпозиції першого А1 блоку

При декомпозиції другого А2 («Розробка підсистеми») блоку виділені 4 роботи:

«Розробка інтерфейсу підсистеми» – входу у даної роботи немає; управлінням будемо вважати правила та стандарти розробки систем; механізмом є програмне забезпечення; виходом для цієї роботи будемо вважати готовий інтерфейс.

Потрібно розробити зручний та легкий інтерфейс для користування як для викладачем так і для студента.

«Розробка БД підсистеми» – даний блок управляється правилами розробки БД; робота виконується за допомогою програмного забезпечення; виходом є – готова БД.

«Розробка скриптів» – входом є готова БД – вихід попередньої роботи; управлінням є, як і в попередньому блоці, правила розробки скриптів, але додалися правила публікації матеріалів; механізм, як і у всіх роботах, є – програмне забезпечення; вихід – робоча підсистема яка легко взаємодія з основною системою віртуального навчального центру.

«Наповнення БД» – робота має два входи: готова БД, робоча підсистема, тестові питання; керується правилами публікації матеріалів; механізмом є програмне забезпечення. Вихід з даної роботи є готова розроблена підсистема. Діаграма декомпозиції першого А1 («Розробка web-системи») блоку представлена на рис. 4.4.

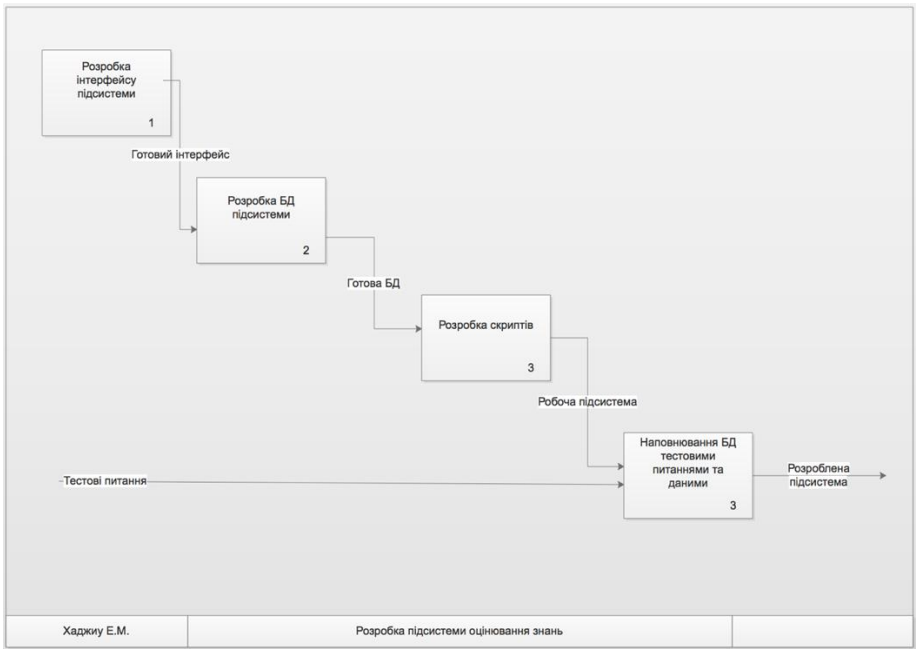


Рисунок 4.4 – Діаграми декомпозиції другого А2 блоку

При декомпозиції третього А3 («Розміщення Web-системи») блоку виділені 3 роботи:

«Настройка web-сервера» – входом для цієї роботи служить Розроблена web-система; управляється правилами настройки Web-сервера та правилами настройки хостингу; механізм – програмне забезпечення; вихід – адрес директорії.

«Загрузка скриптів» – входом для цієї роботи є адрес директорії – вихід попередньої роботи; управлінням є, як і в попередньому блоці, правила розробки скриптів та правила настройки хостингу; механізмом служить програмне забезпечення; вихід – дані авторизації.

«Загрузка БД» – входом є дані авторизації; управління – правила наповнення бази даних; механізмом є програмне забезпечення; вихід роботи: розроблений Rest API та доменне ім’я.

Діаграма декомпозиції другого А3 блоку представлена на рис. 4.5.

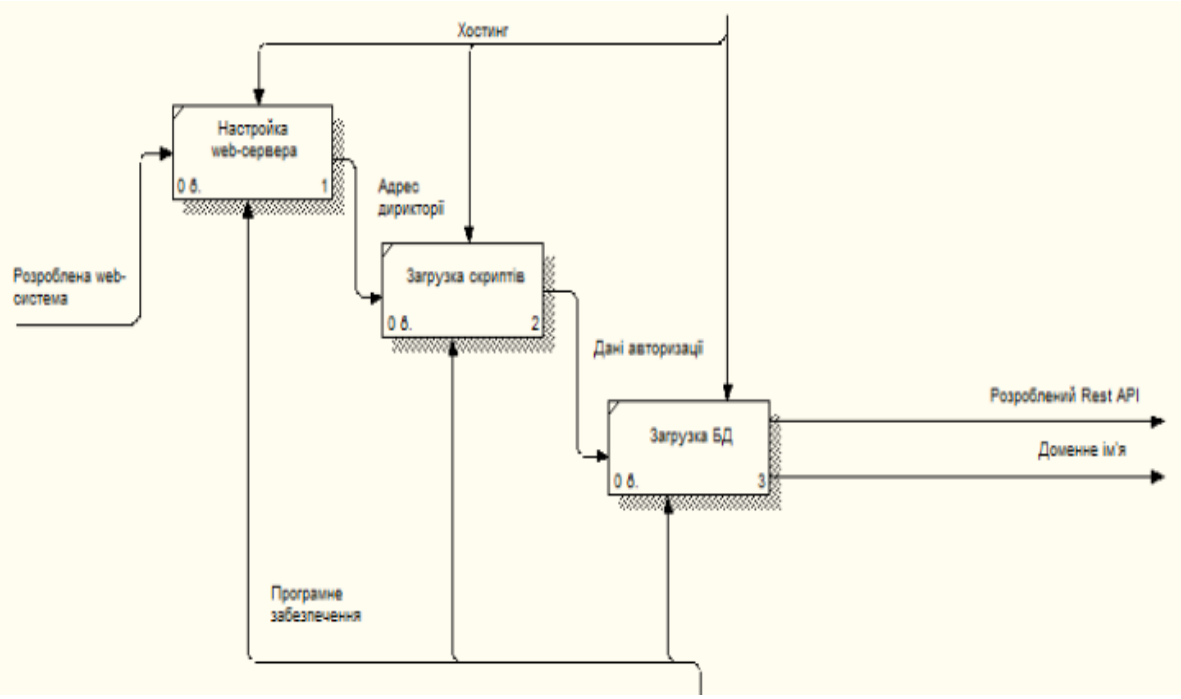


Рисунок 4.5 – Діаграми декомпозиції третього А3 блоку

При декомпозиції четвертого А4 («Розробка додатка») блоку виділені наступні 3 роботи:

«Розробка мобільного інтерфейсу» – входу у даної роботи немає; управлінням будемо вважати правила та стандарти розробки мобільного додатка на платформи Android та iOS; механізмом є програмне забезпечення; виходом для цієї роботи будемо вважати готовий GUI.

«Написання бізнес логіки» – входом для цієї роботи виступає готовий GUI; курується правилами пубілкації матеріалів та правилами написання бізнес логіки для мобільного додатка; механізмом, як у попередній роботі є програмне забезпечення; вихід – готовий мобільний код.

«Розробка мобільної БД» – дана робота має вход попередньої – готовий мобільний код; керується правилами розробки БД та правилами публікації матеріалів; виходом є готовий додаток.

Діаграма декомпозиції четвертого А4 блоку представлена на рис. 4.6.

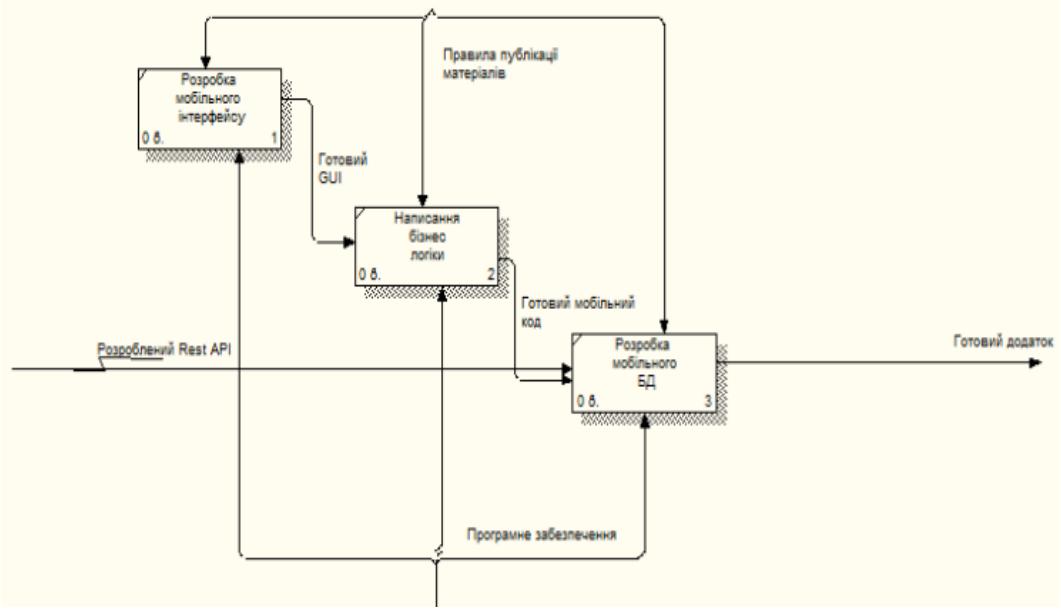


Рисунок 4.6 – Діаграми декомпозиції четвертого А4 блоку

Після того як ми отримаємо готовий додаток відбувається публікація додатка. Після публікації додатка, маємо останній блок А6, при декомпозиції якого виділені 4 роботи:

«Реклама» – ця робота має наступні 3 входи: доменне ім’я, готовий додаток та клієнти які заінтересовані у послугах; дана робота нічим не управляється; механізм – програмне забезпечення; виходом будуть клієнти для за допомогою реклами потрапляють на сайт системи.

«Публікація матеріалів» – робота має наступні входи: клієнт який потрапив на сайт системи через рекламу або напряму та завдання клієнта яке потрібно опублікувати в системі; управлятися робота буде правилами публікації матеріалів; механізму у даної роботи немає; виходом є: завдання клієнтів та навчальні матеріали клієнтів.

«Продаж матеріалів» – входом для цієї роботи служать виходи попередньої, тобто завдання клієнтів та навчальні матеріали клієнтів; управління та механізм у роботи відсутні; виходом є клієнт з навчальним матеріалом та клієнт з готовим завданням, кошти.

«Вивід коштів». Так як головним завданням системи є змога клієнтам отримати репетитерську допомогу, яка оплачується. Вивід коштів є дуже важливою ланкою системи. Входом у роботі є кошти; механізмом є пограмне забезпечення; упраляється правилами платіжної системи у якій здійснюється вивід коштів; виходом є клієнтська монетізація. Діаграма дкомпозиції шстого А6 («Реалізація».) блоку представлена на рис. 4.7.

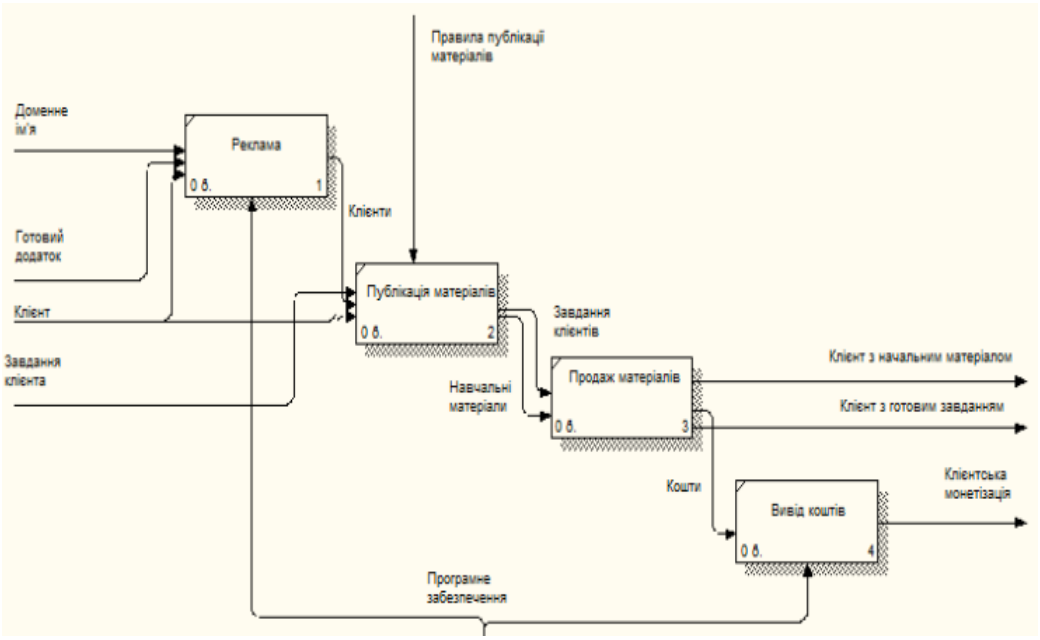


Рисунок 4.7 – Діаграми декомпозиції шостого А6 блоку

4.2Проектування функціональних вимог і БД системи

Для представлення логічної структури інформаційної системи віртуального навчального центру обрана модель представлення даних «сутність-зв'язок». Модель «сутність-зв'язок» ґрунтується на важливої семантичної іформації про реальний світ і призначена для логічного представлення даних. Вона визначає значення даних в контексті їх взаємозв'язку з іншими даними.

Сутність (entity) – це об'єкт, який може бути ідентифікований якимсь способом, що відрізняє його від інших об'єктів. Сутність фактично являє собою безліч атрибутів, які описують властивості всіх членів даного набору сутностей. Безліч значень (область визначення) атрибуту називається доменом. Зв'язок – це асоціація, встановлена між кількома сутностями.

Для бази даних системи виділимо наступні сутності: «User» («Зареєстрований клієнт»), «User_role» («Права клієнта»), «Role» («Списки привелегій»), «Test_role» («Для яких привелеїв доступний тест»), «Tests» («Інформація про тести»), «Test_settings» («Інформація про налаштування тестів»), «Test_comment» («Інформація про те, яким налаштуванням відповідають які коментарі»), «Comment» («Коменрарій про результат проходження тесту»), «User_comment» («Коментарій клієнта»), «Session» («Сесія користувача»), «Results» («Результат проходження тестування»), «Test_question», «Question» («Список тестових питань»), «Types» («Список

типу питань»), «Setting» («Налаштування поточного питання»), «Aidqid» («Перелік всіх відповідей до поточного питання»), «Answer» («Правильна відповідь до поточного питання»), «Test_term» («До якого терміну належить який тест»), «Termin» («Список термінів для категоріювання тестів»), «Vidtid» («До якого словнику відноситься який термін»), «Vocabulary» («Категорії термінів»).

Кожна сутність повинна містити атрибут або групу атрибутів, які будуть однозначно ідентифікувати кожен екземпляр сутності. Такий атрибут називають первинним ключем.

При проектуванні інформаційної системи віртуального навчального центру та її підсистеми було розглянуті сутності та їх атрибути.Структура фізичної моделі даних показана в табл. 4.1–4.21.

Сутність «User» («Зареєстрований клієнт ») містить наступні атрибути(табл. 4.1): uid*, FIO, Login, Pass, E_mail.

Таблиця 4.1 – Сутність «User» («Зареєстрований клієнт »)

№	Атрибут	Тип	Призначення
1	uid*	unsignen int (11)	Код користувача
2	FIO	varchar (50)	Прізвище, ім’я, по-батьвокі кстувача
3	Login	varchar (16)	Логін користувача
4	Pass	varchar (16)	Пароль
5	E_mail	varchar (50)	Електронна адреса

Сутність «User_role» («Права клієнта»),містить наступні атрибути(табл. 4.2): Uid*, Rid*.

Таблиця 4.2 – Сутність «User_role» («Права клієнта»)

№	Атрибут	Тип	Призначення
1	Uid*	unsigned int (11)	Код користувача
2	Rid*	unsigned int (11)	Код прав доступу

Сутність «Role» («Списки привелегій»),містить наступні атрибути(табл. 4.3): name, Rid*.

Таблиця 4.3 – Сутність «Role» («Списки привелегій»)

№	Атрибут	Тип	Призначення

1	name	varchar (50)	Назва
2	Rid*	unsigned int (11)	Код прав доступу

Сутність «Test_role» («Для яких привелеїв доступний тест»), містить наступні атрибути(табл. 4.4): Tid*, Rid*.

Таблиця 4.4 – Сутність «Test_role» («Для яких привелеїв доступний тест»)

№	Атрибут	Тип	Призначення
1	Tid*	unsigned int (11)	Код тесту
2	Rid*	unsigned int (11)	Код прав доступу

Сутність «Tests» («Інформація про тести»), містить наступні атрибути(табл. 4.5): Tid*, Title, Body, Tsid.

Таблиця 4.5 – Сутність «Tests» («Інформація про тести»)

№	Атрибут	Тип	Призначення
1	Tid*	unsigned int (11)	Код тесту
2	Title	varchar(255)	Назва тесту
3	Body	text	Опис тесту
4	Tsid	unsigned int (11)	Код налаштувань

Сутність «Test_settings» («Інформація про налаштування тестів»), містить наступні атрибути(табл. 4.6): Tsid*, Random, Bin_setting, View_res, View_good_res, Count_try, Date_start, Date_stop, Percent, P_good, P_bad.

Таблиця 4.6 – Сутність«Test_settings» («Інформаціяпроналаштуваннятстів»)

№	Атрибут	Тип	Призначення
1	Tsid*	unsignen int (11)	Код налаштування
2	Random	unsignen int (11)	Випадковий порядок для питань
3	Bin_setting	unsignen int (11)	Налаштування тестування
4	View_res	unsignen int (11)	Відображення відповідей
5	View_good_res	unsignen int (11)	Відображення правильних відповідей
6	Count_try	unsignen int (11)	Кількість спроб
7	Date_start	date	Дата початку тестування
8	Date_stop	date	Дата закінчення тестування
9	Percent	unsignen int (11)	Відсоток проходження тесту

10	P_good	text	Текст при позитивному результаті
11	P_bad	text	Текст при негативному результаті

Сутність «Test_comment» («Інформація про те, яким налаштуванням відповідають які коментарі наведені»), містить наступні атрибути(табл. 4.7): Tsid*, Tcid*.

Таблиця 4.7 – Сутність «Test_comment» («Інформація про коментарі»)

№	Атрибут	Тип	Призначення
1	Tsid*	unsignen int (11)	Код налаштування
2	Tcid*	unsignen int (11)	Код коментарів

Сутність «Comment» («Коменрарій про результат проходження тесту»), містить наступні атрибути(табл. 4.8): Tcid*, Percent_min, Body, Percent_max, title.

Таблиця 4.8 – Сутність «Comment» («Коментарій про результат »)

№	Атрибут	Тип	Призначення
1	Tcid*	unsignen int (11)	Код коментаря
2	Percent_min	unsignen int (11)	Мінімальний відсоток
3	Body	text	Текст оцінки
4	Percent_max	unsignen int (11)	Максимальний відсоток
5	title	varchar(50)	Назва оцінки

Сутність «User_comment» («Коментарій клієнта»), містить наступні атрибути(табл. 4.9): Tcid*, Uid*.

Таблиця 4.9 – Сутність «User_comment» («Коментарій клієнта»)

№	Атрибут	Тип	Призначення
1	Tcid*	unsignen int (11)	Код коментарів
2	Uid*	unsignen int (11)	Код користувача

Сутність «Session» («Сесія користувача»), містить наступні атрибути(табл. 4.10): Sid*, Uid, Session.

Таблиця 4.10 – Сутність «Session» («Сесія користувача»)

№	Атрибут	Тип	Призначення

1	Sid*	unsignen int (11)	Код сесії
2	Uid	unsignen int (11)	Код користувача
3	Session	text	Інформація про сесію

Сутність «Results» («Результат проходження тестування»), містить наступні атрибути(табл. 4.11): Qid*, Uid*,Tid*, Percent, Ball.

Таблиця 4.11 – Сутність «Results» («Результат проходження тестування»)

№	Атрибут	Тип	Призначення
1	Qid*	unsignen int (11)	Код питання
2	Uid*	unsignen int (11)	Код користувача
3	Tid*	unsignen int (11)	Код тестування
4	Percent	unsignen int (11)	Процент
5	Ball	unsignen int (11)	Оцінка у балах

Сутність «Test_question», містить наступні атрибути(табл. 4.12): Qid*, Tid*. Таблиця містить список відповідності між тестами і питаннями.

Таблиця 4.12 – Сутність «Test_question»

№	Атрибут	Тип	Призначення
1	Qid*	unsignen int (11)	Код питання
2	Tid*	unsignen int (11)	Код тестування

Сутність «Question» («Список тестових питань»), містить наступні атрибути(табл. 4.13): Qid*, Body, Sid, Tid. Таблиця містить список питань.

Таблиця 4.13 – Сутність «Question» («Список тестових питань»)

№	Атрибут	Тип	Призначення
1	Qid*	unsignen int (11)	Код питання
2	Body	text	Текст питання
3	Sid	unsignen int (11)	Налаштування питання
4	Tid	unsignen int (11)	Тип питання

Сутність «Types» («Список типу питань»), містить наступні атрибути(табл. 4.14): Tid*, Sid, Title. Таблиця містить список типів питань.

Таблиця 4.14 – Сутність «Types» («Список типу питань»)

№	Атрибут	Тип	Призначення
1	Tid*	unsignen int (11)	Код типу питання
2	Sid	unsignen int (11)	Код налаштування
3	Title	varchar (50)	Заголовок

Сутність «Setting» («Налаштування поточного питання»), містить наступні атрибути(табл. 4.15): Sid*, Multy, Random. Таблиця містить налаштування до поточного питання.

Таблиця 4.15 – Сутність «Setting» («Налаштування поточного питання»)

№	Атрибут	Тип	Призначення
1	Sid*	unsignen int (11)	Код налаштування
2	Multy	unsignen int (11)	Множинний вибір відповідей
3	Random	unsignen int (11)	Випадковий порядок для відповідей

Сутність «Aidqid» («Перелік всіх відповідей до поточного питання»), містить наступні атрибути(табл. 4.16): Aid*, Qid. Таблиця зберігає всі відповіді до поточного питання.

Таблиця 4.16 – Сутність «Aidqid» («Перелік всіх відповідей до поточного питання»)

№	Атрибут	Тип	Призначення
1	Aid*	unsignen int (11)	Код питання
2	Qid	unsignen int (11)	Код відповіді

Сутність «Answer» («Правильна відповідь до поточного питання»), містить наступні атрибути(табл. 4.17): Aid*, Body, Answer. Таблиця містить в собі правильну відповідь.

Таблиця 4.17 – Сутність «Answer» («Правильна відповідь до поточного питання»)

№	Атрибут	Тип	Призначення
1	Aid*	unsignen int (11)	Код відповіді

2	Body	Text	Текст відповіді
3	Answer	unsignen int (11)	Маркер правильної відповіді

Сутність «Test_term» («До якого терміну належить який тест»), містить наступні атрибути(табл. 4.18): Test_id*, Term_id*. Таблиця містить в собі інформацію про те, до якого терміну належить який тест.

Таблиця 4.18 – Сутність «Test_term» («До якого терміну належить який тест»)

№	Атрибут	Тип	Призначення
1	Test_id*	unsignen int (11)	Код тексту
2	Term_id*	unsignen int (11)	Текст терміну

Сутність «Termin» («Список термінів для категоріювання тестів»), містить наступні атрибути(табл. 4.19): Tid*, Title. Таблиця зберігає список термінів для категоріювання тестів.

Таблиця 4.19 – Сутність «Termin» («Список термінів для категоріювання тестів»)

№	Атрибут	Тип	Призначення
1	Tid*	unsignen int (11)	Код терміну
2	Title	varchar(50)	Заголовок

Сутність «Vidtid» («До якого словнику відноситься який термін»), містить наступні атрибути(табл. 4.20): Vid*, Tid*. Таблиця містить інформацію про те, до якого словника відноситься який термін.

Таблиця 4.20 – Сутність «Vidtid» («До якого словнику відноситься який термін»)

№	Атрибут	Тип	Призначення
1	Vid*	unsignen int (11)	Код словника
2	Tid*	unsignen int (11)	Код терміну

Сутність «Vocabulary» («Категорії термінів»), містить наступні атрибути(табл. 4.21): Vid*, Title.

Таблиця 4.21 – Сутність «Vocabulary» («Категорії термінів»)

№	Атрибут	Тип	Призначення
1	Vid*	unsignen int (11)	Код словника
2	Title	varchar(50)	Заголовок

Описавши, всі головні сутності й атрибути розроблюваної бази даних для інформаційної системи віртуального навчального центру можна визначити зв'язки між сутностями. Уявімо базу даних у вигляді моделі «сутність-зв'язок».

В багатотабличних базах даних дані зберігаються в різних таблицях. Між таблицями повинні бути встановлені зв'язки. Зв'язки встановлюють на основі даних в співпадаючих полях – по змісту та по типу даних. Зв'язані поля можуть мати різні імена. Одне із зв'язаних полів повинно бути ключовим. Існують 4 типи зв'язків:

- один-до-одного;
- один-до-багатьох;
- багато-до-одного;
- багато-до-багатьох.

У інформаційній системі віртуального навчального центру здійснюється відношення «один-до-багатьох». Відношення «один-до-багатьох» досить широко поширені. При побудові інтерфейсів це відношення також називають «головний-детальний» (master-detailed) (головний – «один», детальний – «багато»). Діаграма «сутність-зв'язок» представлена на рис. 4.8.

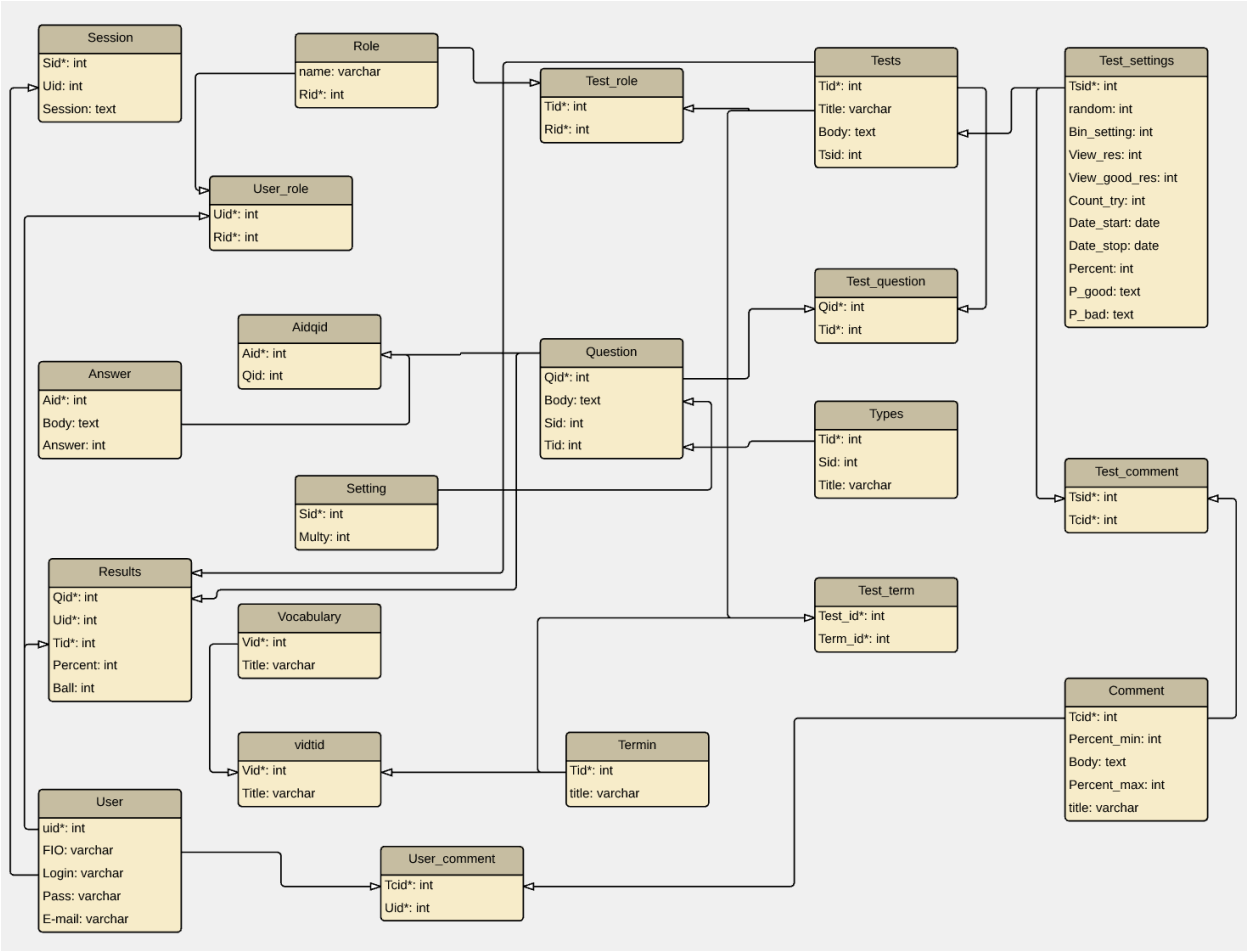


Рисунок 4.8 – Модель бази даних «сутність-зв'язок»

4.3 Опис процесів підсистеми планування проведення та оцінювання знань

Однією з головних цілей підсистеми планування проведення та оцінювання знань є зручність для користувача у серфінгу системою. Користувач у системі може бути як звичайним учнем, так і педагогом якому надаються права на проведення тестування у підсистемі.

Перед тим як мдераторові системи надати той чи інший статус користувачеві, необхідна реєстрація клієнта. Тільки після реєстрації, адміністратор системи може надати клієнтові статус педагога(викладача).Процес реєстрації наведений на рис. 4.9.

Рисунок 4.9 – Реєстрація користувача в підсистемі перевірки та оцінювання знань

Після вдалої реєстрації клієнта, він може скористатися усіма привілеями системи навчального центру, він може розміщати навчальні матеріали, задавати питання та потім отримати відповіді від інших користувачів, вводити або виводити кошти отримані після продажу своїх публікацій.

Головним додатком підсистеми є те що відтепер користувачеві надається можливість перевірити свої знання пройшовши тестування підготовлене викладачем. Тестування можна пройти або після співпраці з викладачем, або ж виключно для перевірки своїх базових знань. Процес тестування представлений на рис. 4.10.

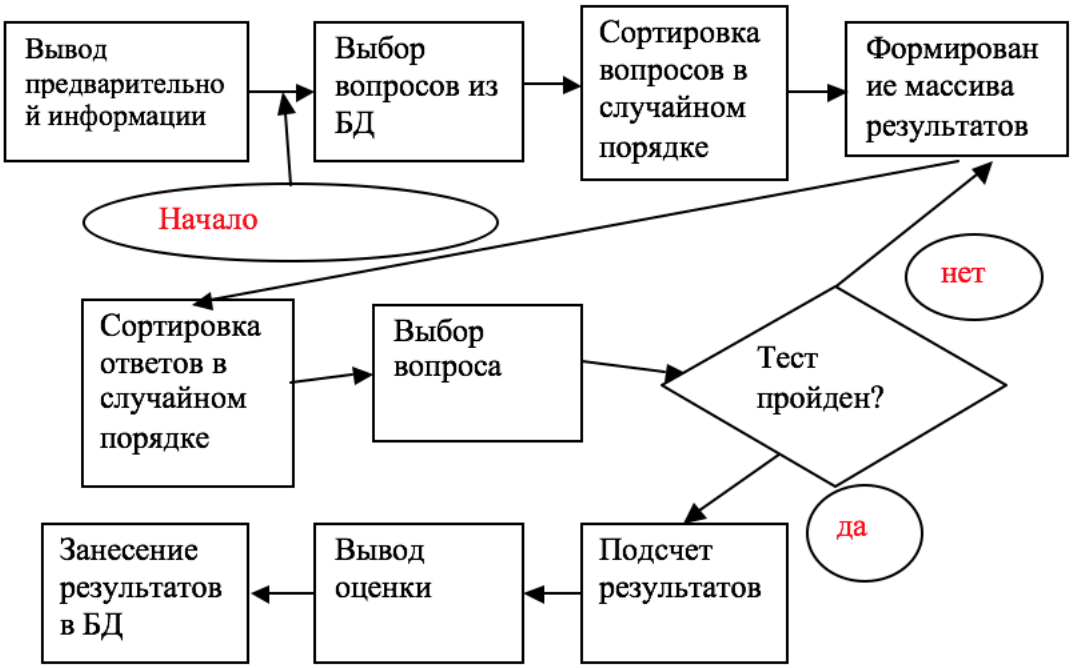


Рисунок 4.10 – Проходження тестування у підсистемі

Після того як учень(користувач) пройшов тестування йому повідомляється оцінка, усі дані також передаються до викладача. Викладач має змогу перевіряти результати, сортувати користувачів за оцінкою, швидкістю написання тесту, тощо. Процес формування статистики представлений на рис. 4.11.

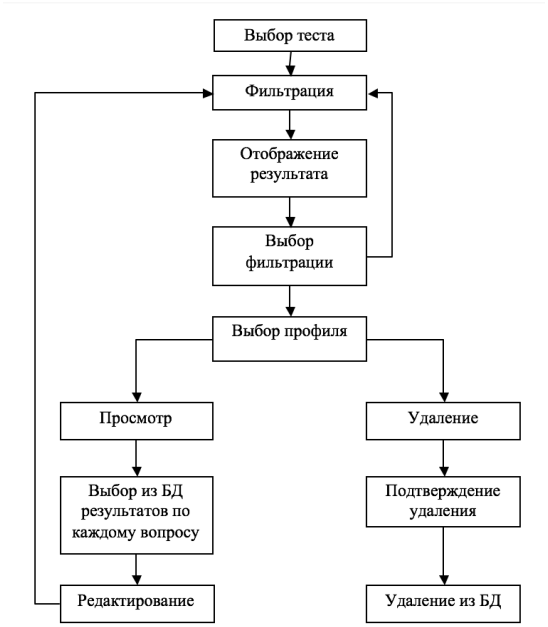


Рисунок 4.11 – Процес формування статистики для викладача

4.4 Проектування служб для користувача інтерфейсу

Навігація по інформаційній системі – це той механізм, який дозволяє відвідувачеві знайти потрібну йому інформацію. Він спирається на логічну структуру і допомагає користувачеві швидко по ній переміщатися. Ефективність навігації можна оцінити правилом «трьох натискань», яке полягає в тому, що до будь-якого документу, що знаходиться в Web- системі, можна потрапити з головної сторінки, перейшовши не більше ніж за трьома посиланнями.

Для інформаційної підсистеми планування проведення та оцінювання знань передбачено 2 види користувача: користувач-учень і користувач-викладач. Вони будуть мати різні функції і привілеї.

Зайшовши в систему користувач-клієнт через браузер має змогу здійснювати авторізацію або реєстрацію. Для цього йому потрібно меню «Сторінка реєстрації» або «Сторінка авторизації».

Зайшовши в систему користувач-клієнт через браузер має змогу здійснювати авторізацію або реєстрацію. Для цього йому потрібно меню «Сторінка реєстрації» або «Сторінка авторизації».

При перегляді меню «Завантаження питань» користувачу-клієнту надається доступ до завантаження питання.

Перейшовши в меню «Завантаження навчальних матеріалів» клієнт може завантажити свій готовий навчальний матеріал.

Меню «Основні завантажені питання» та «Основні завантажені навчальні матеріали» показують найпопулярніші завантажені питання та нвчальні матеріали.

При перегляді меню «Про Нас» клієнтові надається інформація про сстему, останні новини.

При перегляді меню «Мій аккаунт» клієнтові надається доступ до своїх загружених навчальних матеріалів на питань, доступ до куплених та проданих навчальних матеріалів, доступ до балансу. У аккаунті користувач-учень може бачити інформацію про пройдене тестування у одного з користувачів-викладачів.

Клієнт має змогу здійснити «Пошук» у системі.

Нижче представлена логічна структура системи для веб-користувача у загальній системі віртуального навчального центру. Червоні блоки, це блоки які були додані у виконанні підсистеми оцінювання знань (рис. 4.12).

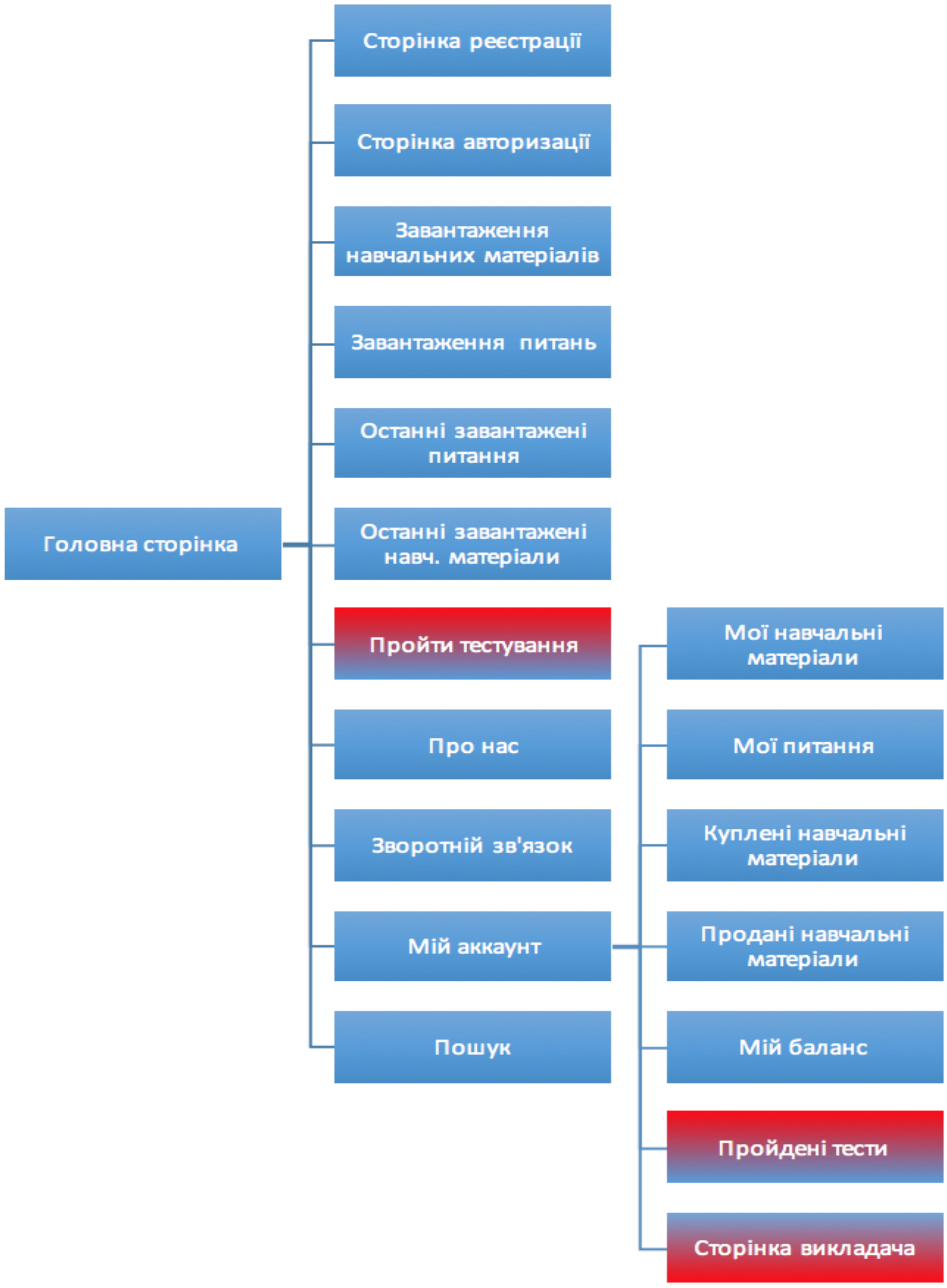


Рисунок 4.12 – Логічна структура інформаційної системи для користувача-клієнта

Зайшовши в систему користувач-клієнт має змогу здійснювати авторізацію або реєстрацію. Також надається можливість перегляду всіх куплених та опублікованих навчальних матеріалів та топ 10 матеріалів в інформаційній системі. Користувач може переглянути інформацію про аккаунт та здійснити пошук серед навчальних матеріалів та здійснити фінансову операцію купівлі матеріалу з можливістю завантажити файли на мобільний пристрій або персональний компьютер. Для проходження тестування реєстрація клієнта обов’язкова.

Для Web-системи дуже важливий стиль, що додає Інтернет-ресурсу власне обличчя і пізнаваність. Можна виділити наступні елементи, що у створенні стилю:

– шрифт – в межах публікації він повинен мати однакові характеристики (накреслення, висоту, колір);
– колірна схема Wеb-сторінок – вибір тих трьох кольорів сторінки, котрі будуть використовуватися для представлення звичайного тексту, посилань і відвіданих посилань. Колірна схема повинна повторюватися на всіх сторінках публікації, це створить у відвідувача відчуття зв'язності системи. Кольори посилань вибирають таким чином, щоб вони були помітні і в той же час не заважали читати основний текст;
– графічне оформлення системи – має укладатися в загальну колірну схему.

Результатом цього етапу проектування повинен бути остаточний ескіз Wеb-сторінки.

Професійна розробка відрізняється від аматорської насамперед тим, що всі сторінки Інтернет-системи виконані в єдиному стилі. Щоб витримати стиль, необхідно на початку розробити шаблон сторінки. Шаблони зручні тим, що більшість сторінок верстають за подобою однієї сторінки майже автоматично. Основні елементи, які повинні бути присутніми на сторінках інформаційної системи фармацевтичної фірми:

– назва;
– логотип (графічний знак, який ідентифікує компанію);
– навігаційне меню;
– дані (власне зміст сторінки).

Перш за все, слід приділити увагу сервісному обслуговуванню – зручний каталог продукції, пошук по товарах і послугах, ясні і не складні форми, які необхідно заповнити користувачеві для реєстрації, формуванні змовлення та покупки, контактна інформація. Графічні елементи використовуються не тільки ілюстрації, фотографії продукції, але піктограми, іконки – графічні символи, які допомагають відвідувачеві орієнтуватися в інформаційних обсягах.

Структура шаблону складається з наступних елементів: назва системи; основні сторінки системи; поле реєстрації та авторизації; поле інформації; поле в якому надається можливість задати нове питання; останні завантажені навчальні матеріали та останні завантажені питання; поле топ навчальні

матеріали та топ вчителів та форма пошуку. У нижній частині системи розташується підвал системи (рис. 4.13).

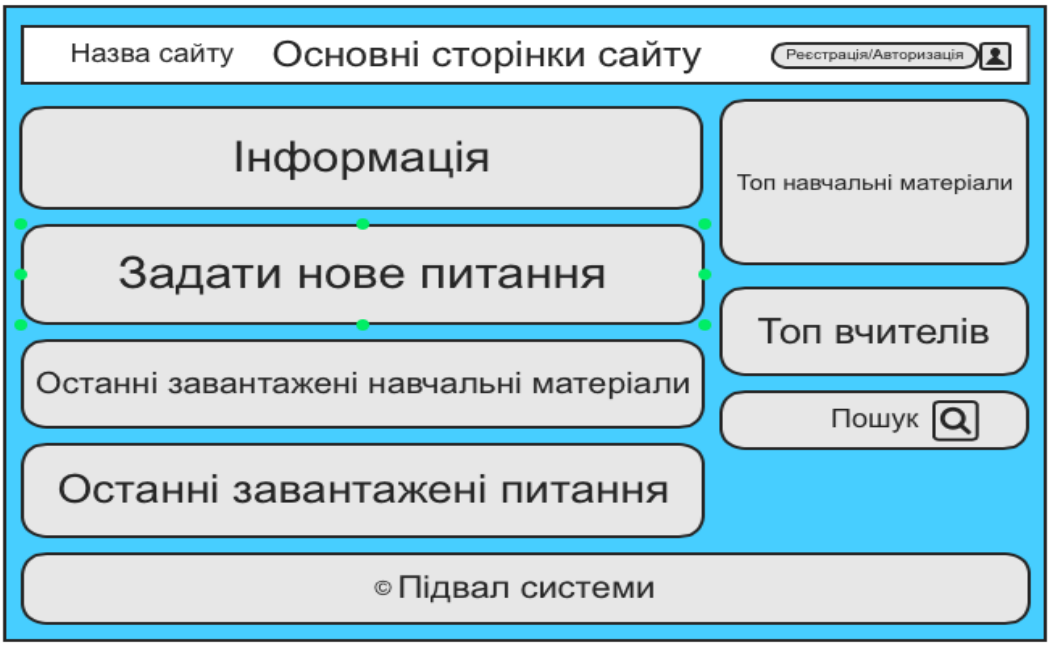


Рисунок 4.13 – Структура головної сторінки інформаційної системи віртуального навчального центру та підсистеми планування та оцінювання знань

5 ПРАКТИЧНА РЕАЛІЗАЦІЯ ЗАСТОСУВАННЯ КОРИСТУВАЧА

5.1 Керівництво користувача

Перше уявлення про систему дає домашня сторінка – ця сторінка відображається при першому вході в інформаційну систему користувача. Головна сторінка проектується за допомогою CMS (системи управління контентом). З огляду на функцію головної сторінки, вона повинна бути оформлена так, щоб користувач зміг розібратися в навігації сайту, ближче ознайомитися з ресурсом і захотів перейти на наступні сторінки. Тому головна сторінка буває дочиста просто напхана корисними інструментами, які полегшують пошук необхідної інформації. Але все ж найголовніше – це текстова складова. Матеріал на головній сторінці повинен бути побудований таким чином, щоб в лічені секунди зацікавити користувача і потім змусити вчинити будь-яку дію, наприклад, зробити покупку.

У верхній частині сторінки розміщена назва системи, що б повідомити користувачеві профіль, призначення та діяльність системи, також у верхній частині сторінки розташоване загальне меню системи. Користувачу-клієнту буде цікаво дізнатися про найбільш продавані товари, саме для цього на головній сторінці, у правій частині розташований список лідерів продаж з різних категорій, лідери серед вчителів та також пошук в системі.

Для того аби пройти тестування, користувачу буде показана на верхньому меню кнопка «Testing», натиснувши на неї, незареєстрований користувач перейде на сторінку реєстрації, а зареєстрований користувач зможе обрати допустимі на даний час тести різних викладачів.

В середині сторінки розташований блок для розміщення нового питання, останні задані питання та останні розміщенні навчальні матеріали. У нижній частині системи розташований футер або підвал системи де рзташований копірайт системи.

На головній сторінки представлені два функціональних меню – горизонтальне і вертикальне. У горизонтальному меню можна переглянути дані про систему, всю необхідну контактну інформацію та переглянути свій рунок та аккаунт. Домашня сторінка Інтернет-системи представлена на рис. 5.1.

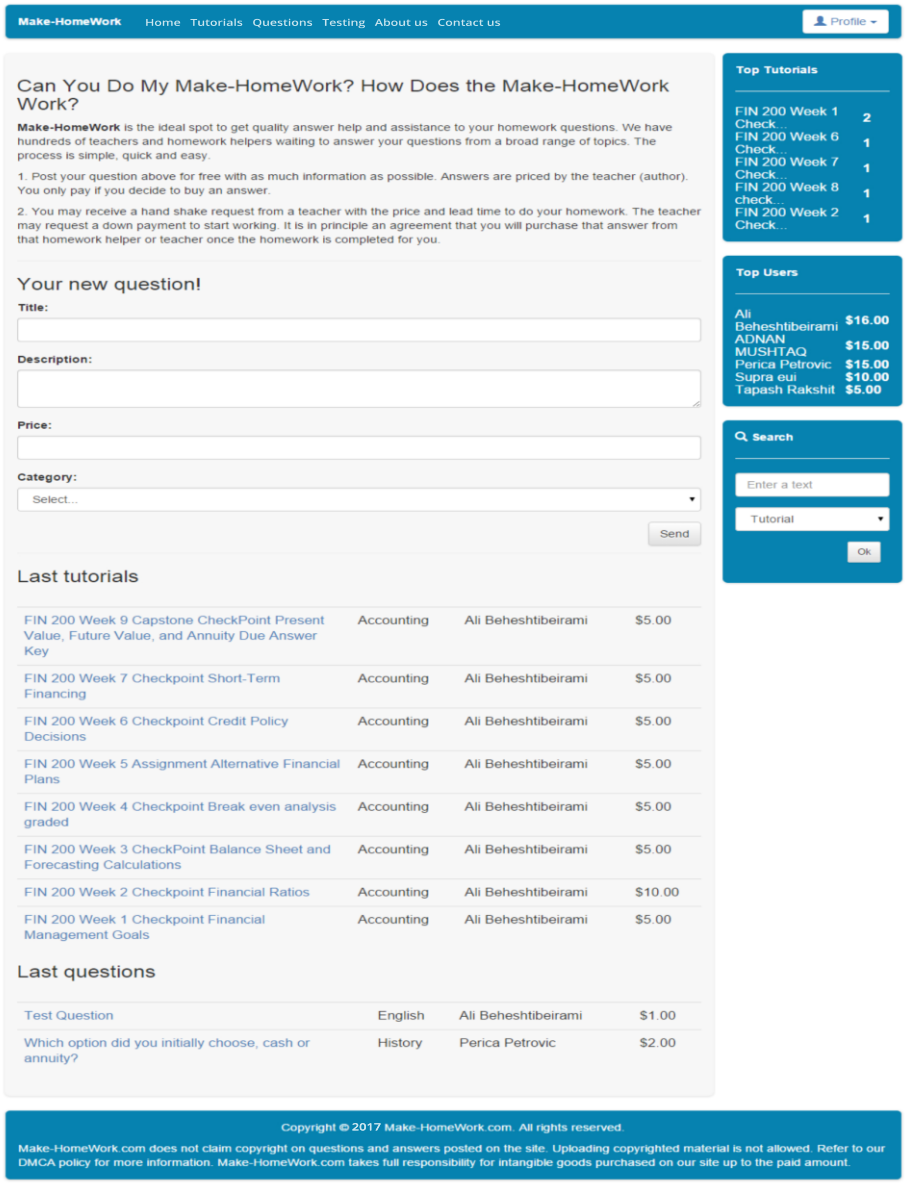


Рисунок 5.1 – Головна сторінка інформаційної системи віртуального навчального центру з підсистемою планування та оцінювання знань

Користувачу надається змогу реєстрації або ж авторизації. Це можна зробити перейшовши з основного меню до сторінок реєстрації або авторизації, дані сторінки представлені на рис. 5.2 та рис. 5.3.

Для реєстрації користувачу потрібно надати наступні дані: email, пароль та ім’я та фамілію.

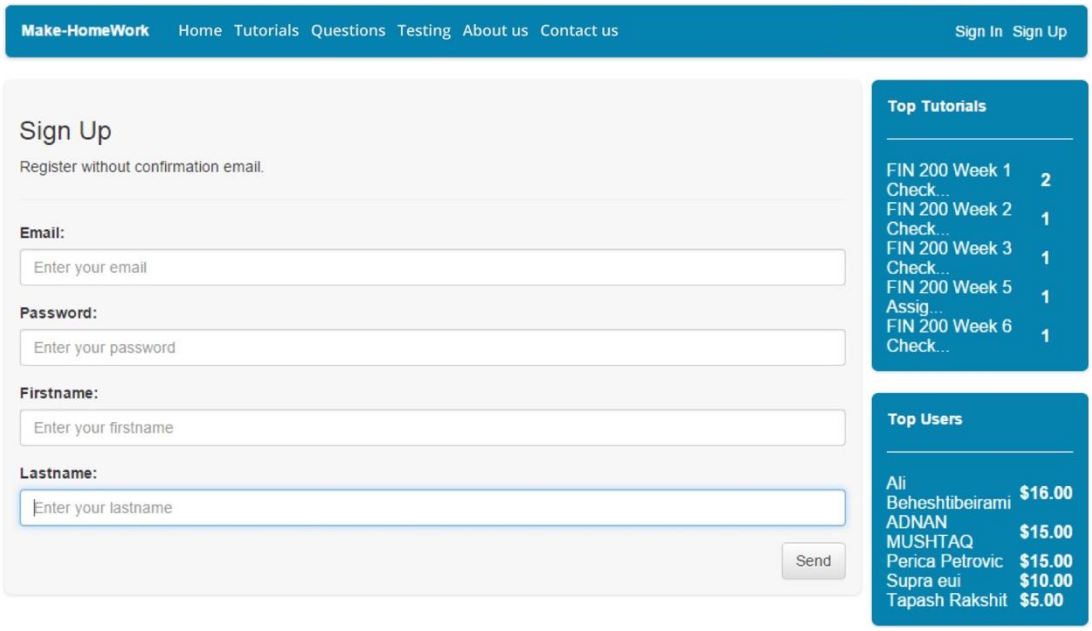


Рисунок 5.2 – Сторінка реєстрації користувача-клієнта

Після реєстрації користувач може увійти у свій профіль. Для цього необхідно ввести email та пароль.

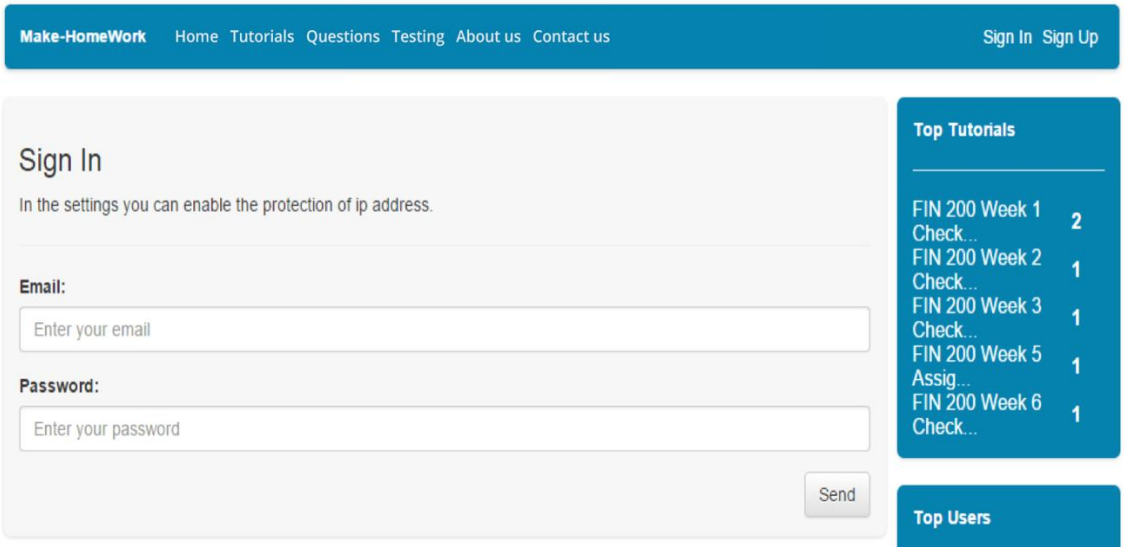


Рисунок 5.3 – Сторінка авторизації користувача-клієнта

Після авторизації користувачеві надається доступ до особистого профіля у якому він може переглянути свої продані навчальні матеріали, свої завантажені навчальні матеріали та питання. Також надається змога відразу перейти на сторінку завантаження питання або питання. Надається можливість поповнення рахунку або ж вивід коштів, перегляд історії рахунка. Надається доступ до параметрів профіля «My settings» (рис 5.4) У профілі надається змога перейти до завантажених тестів(якщо користувач є виклада-

чем). Також можна перейти до профіля викладача. Права викладача надаються адміністратором системи.

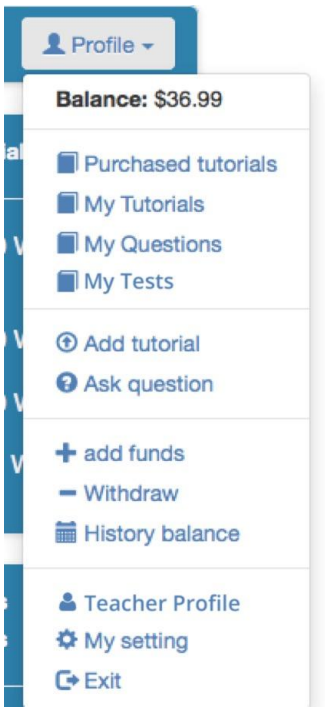


Рисунок 5.4 – Профіль користувача-викладача

5.2 Керівництво користувача-викладача

Після перегляду профіля користувач може перейти до перегляду профіля викладача(який доступний і для інших клієнтів). У профілі викладача надається така інформація про викладача як: ім’я викладача, вік, місце проживання, рейтинг від клієнтів, дата реєстрації, остання дата авторизації в систему.

Клієнт звертає не аби яку увагу і на наступні критерії: улюблені категорії викладача, дані про себе, та доступні на даний час для проходження тестування. Клієнти можуть побачити чи є доступним даний викладан наразі у системі, та може написати йому повідомлення. Також того або іншого викладача можна додадити до своїх обраних викладачів.

Саме тут, у профілі викладача, користувач-студент може спробувати пройти тестування, тим самим перевіривши свої знання у той або інший галузі. Сторінку профіля викладача представлена на рис. 5.5.

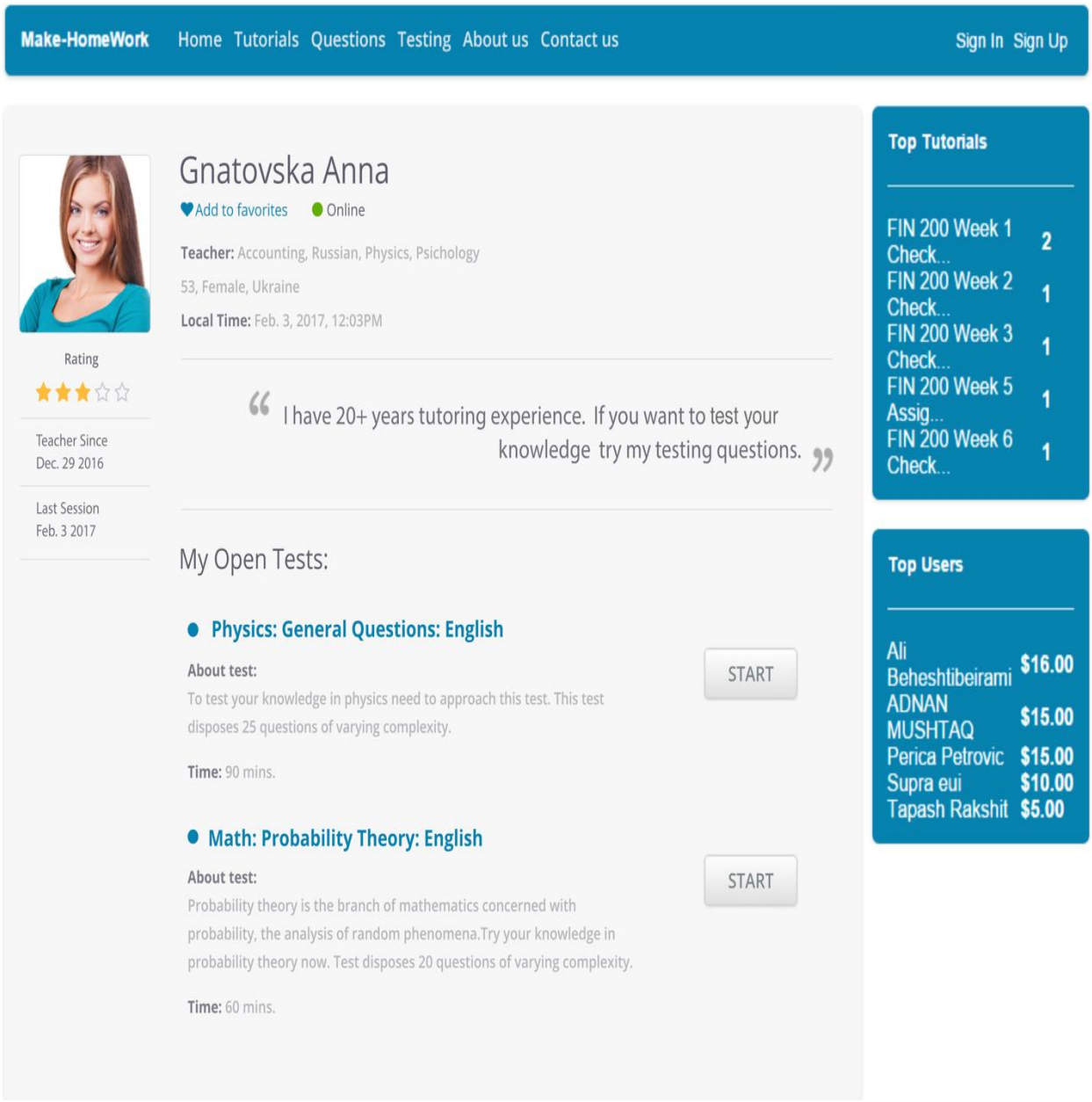


Рисунок 5.5 – Сторінка користувача-викладача

Для того аби викладачеві добавити тест до свого профіля, слід пройти на сторінку додавання тестування у систему. Тут викладач обирає ім’я тестування та категорію спеціальності та дисципліни до якої буде додане тестування.Також додається опис тестування, яке допоможе студенту зрозуміти складність та інші данні про тестування. Тільки після того як усі налаштування будуть додані можна буде перейти до наступного кроку додавання питань до тесту. Процес додавання тестування до системи представлено на рис. 5.6.

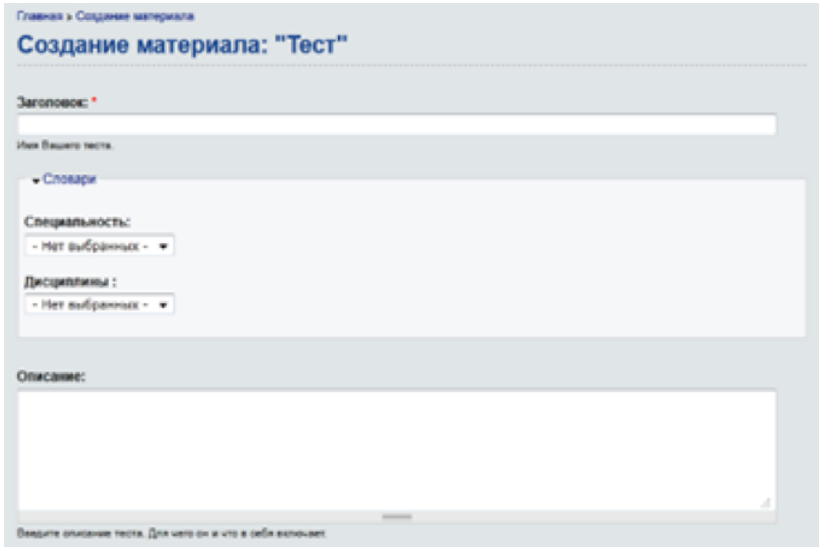


Рисунок 5.6 – Сторінка додавання тесту до підсистеми перевірки та оцінювання знань

У підсистемі перевірки та оцінювання знань є можливість для викладача налаштувати процес проходження тестування зручніше для себе.

У налаштуваннях тестування ви можете вибрати наступні пункти:

– відновити тест. Якщо користувач раптом випадково перейшов на іншу сторінку або закрив тестування, він зможе повернутися до того місця, де зупинився, зі збереженням усіх попередніх відповідей;
– дозволити пропустити питання. Цей пункт дозволить пропустити питання в тесті;
– дозволити перемикатися між питаннями. Якщо користувач пропустив складні питання, він завжди зможе до них швидко повернутися за допомогою випадаючого вікна списку питань;
– повторити питання, якщо дали правильну відповідь. Цей пункт можна використовувати для пробного тестування. Якщо обрана дана функція, то система не дозволятиме користувачеві перейти до наступного питання поки не буде дана правильна відповідь на поточне питання;
– сортувати питання у випадковому порядку. Якщо обраний пункт «Випадковий порядок», то всі питання будуть сортуватися випадковим чином.

Якщо вам захочеться показати результати проходження тесту користувачеві, то Ви можете вибрати наступне: показати результати у кінці тестування, показувати правильну відповідь після кожного питання. Налаштування проходження тестування представлено у рис. 5.7.

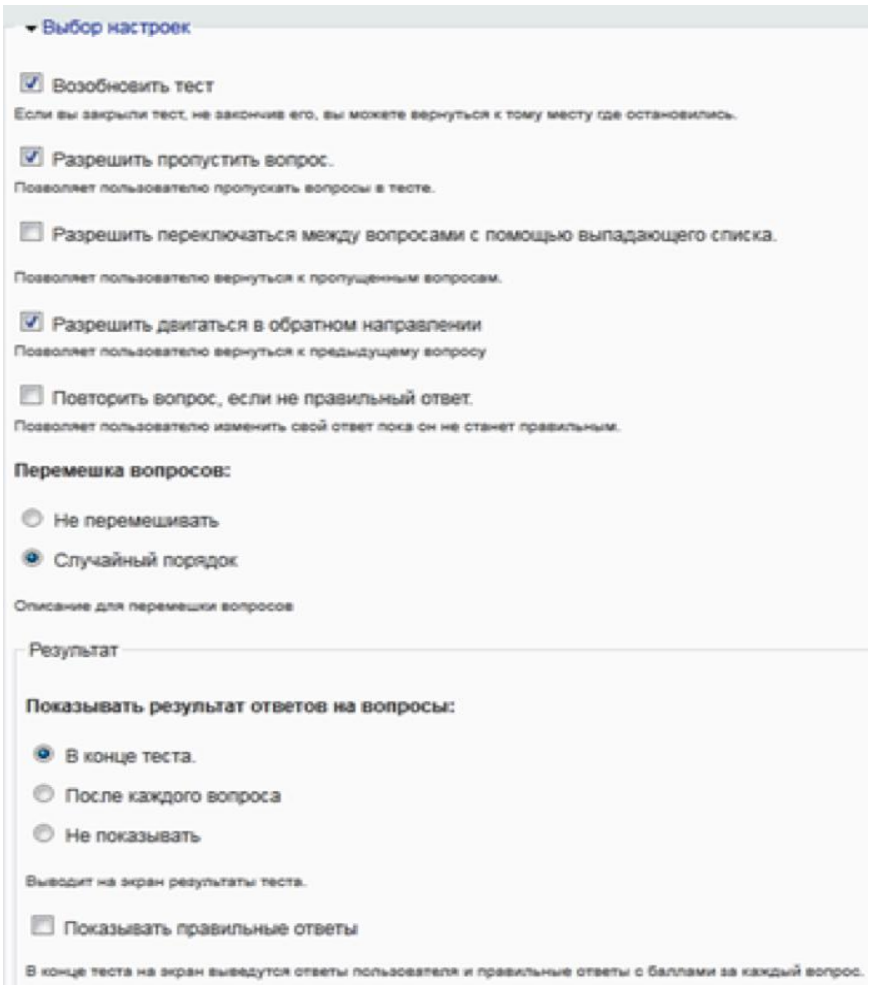


Рисунок 5.7 – Сторінка налаштування проходження тестування у підсистемі оцінювання знань

Існують ще кілька налаштувань для викладача, які дають змогу зручніше організувати процес проходження тестування. На сторінці налаштування тестування можна обрати кількість спроб проходження тесту для одного користувача, які результати зберегти(найкращі, останні або всі).

На сторінці налаштування тестування можна обрати доступність тесту, якщо ви бажаєте щоб даний тест був доступний завжди та всім клієнтам-користувачам, слід додати відмітку “завжди доступний”, якщо бажаєте налаштувати тестування тільки для якогось інтервалу часу, є можливість додати цей інтервал. Налаштування цих критеріїв показана на рис. 5.8.

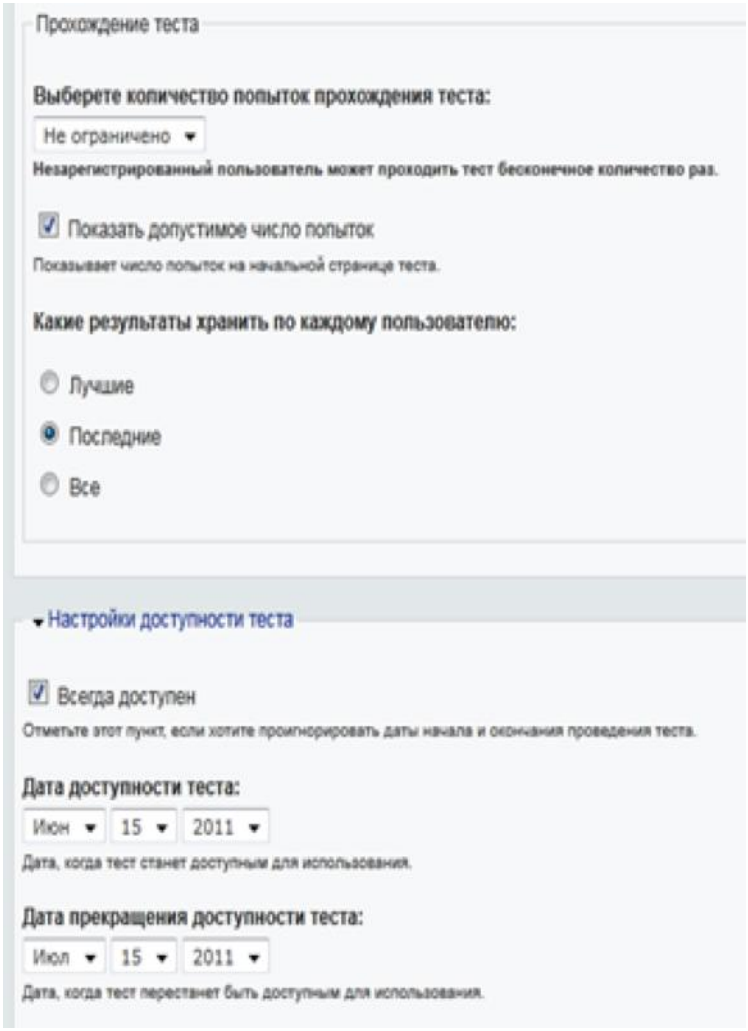


Рисунок 5.8 – Сторінка налаштування проходження тестування у підсистемі оцінювання знань

5.2.1 Створення питань для тестування

Після того як тест був створений можна приступити до створення питань для нього. Для цього необхідно вибрати в меню користувача пункт «Створення матеріалу». Перед вами відкриється форма створення різних типів матеріалів.

Ви можете обрати різні типи питань, які ви б хотіли включити в своє тестування: питання з одним або декількома правильними відповідями, птання з рукописним відповіддю, питання-відповідність, що включає в себе питання-порядок. Приклад створення різних типів питань надано на рис. 5.9.

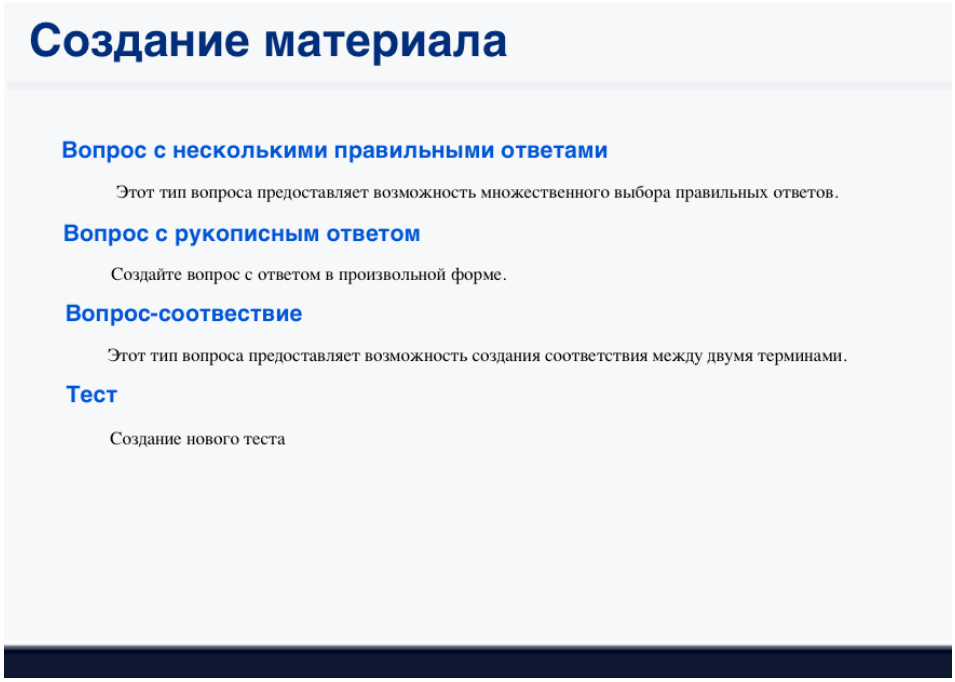


Рисунок 5.9 – Сторінка вибору типу нових питань для тестування

Для створення питання з одним або декількома правильними відповідями необхідно натиснути на напис «Питання з декількома правильними відповідями», після чого перед вами відкриється форма створення даного типу питань(рис. 5.10).

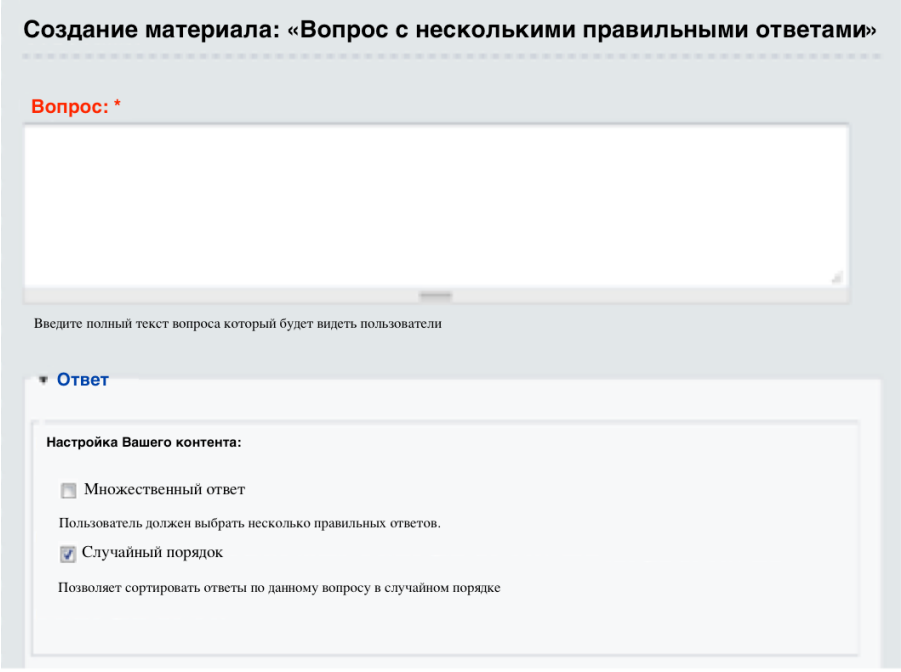


Рисунок 5.10 – Форма додавання нового питання з одним або кількома правильними відповідями

У даній формі ви можете ввести текст свого питання і вибрати настройки для відповіді. Пункт «множинна відповідь» вказує на те, скільки правильних відповідей в даному питанні, один або декілька. Якщо галочки поруч з цим пунктом немає, значить питання з декількома правильними відповідями автоматично стає питанням з однією правильною відповіддю. Пункт «випадковий порядок» дозволяє сортувати відповіді з даного питання у випадковому порядку, що допоможе виключити можливість підказки студентів один одному. Після того, як ви вказали всі налаштування, вам необхідно вказати правильні відповіді для цього питання (рис. 5.11).

Рисунок 5.11 – Форма додавання нового питання з одним або кількома правильними відповідями

Ви можете вказати кількість варіантів відповіді для даного питання, а потім додати питання в уже існуючий тест або в кілька тестів. Для створення питання з рукописною відповіддю вибираємо на сторінці викладача відповідну форму. Процес додавання питання с рукописною відповіддю можна побачити на рис. 5.12.

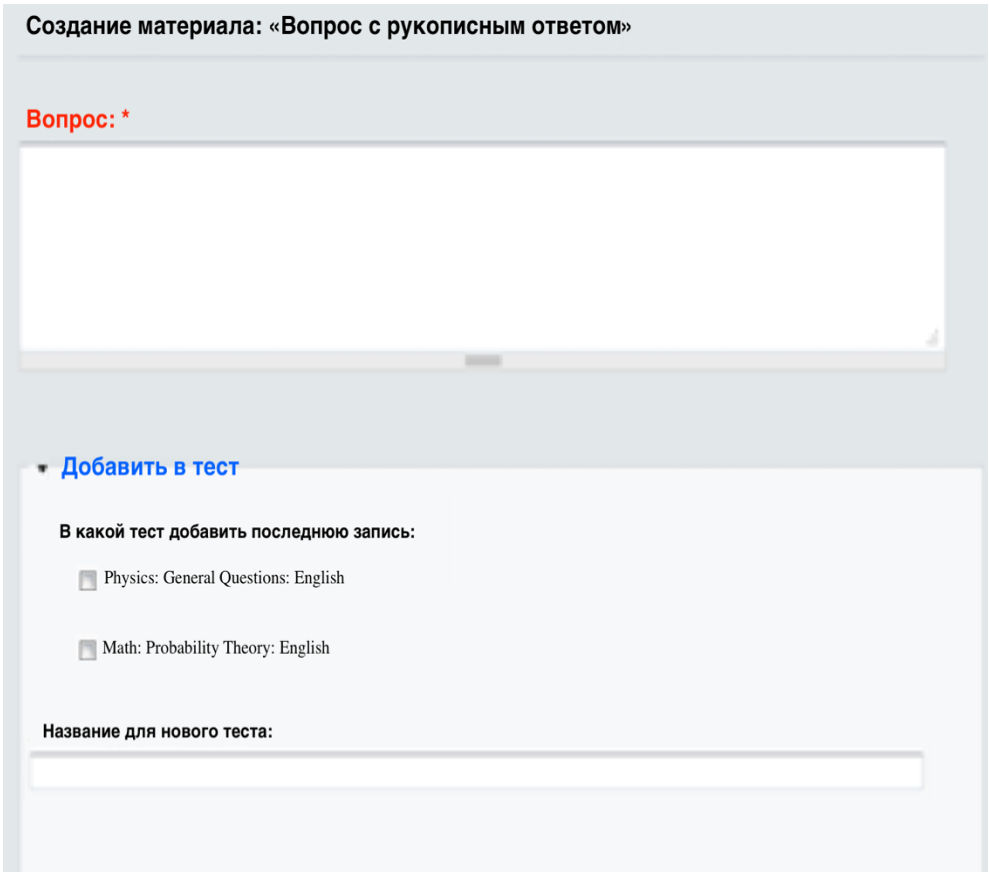


Рисунок 5.12 – Форма додавання нового питання з рукописною відповіддю

У даній формі досить просто ввести текст питання і визначити, до якого тесту буде ставитися це питання. Суть питання з рукописною відповіддю полягає в тому, що після проходження тесту користувачем, викладачеві дедеться перевірити цю відповідь особисто, і поставити відповідний бал. Потім ви можете додати питання в уже існуючий тест або в кілька тестів, або ж створити для даного питання новий тест.

Для створення питання-відповідності вибираємо на сторінці викладача відповідну форму. Перед вами з'явитися форма створення питання-відповідності. Процес додавання питання-відповідності можна побачити на рис. 5.13.

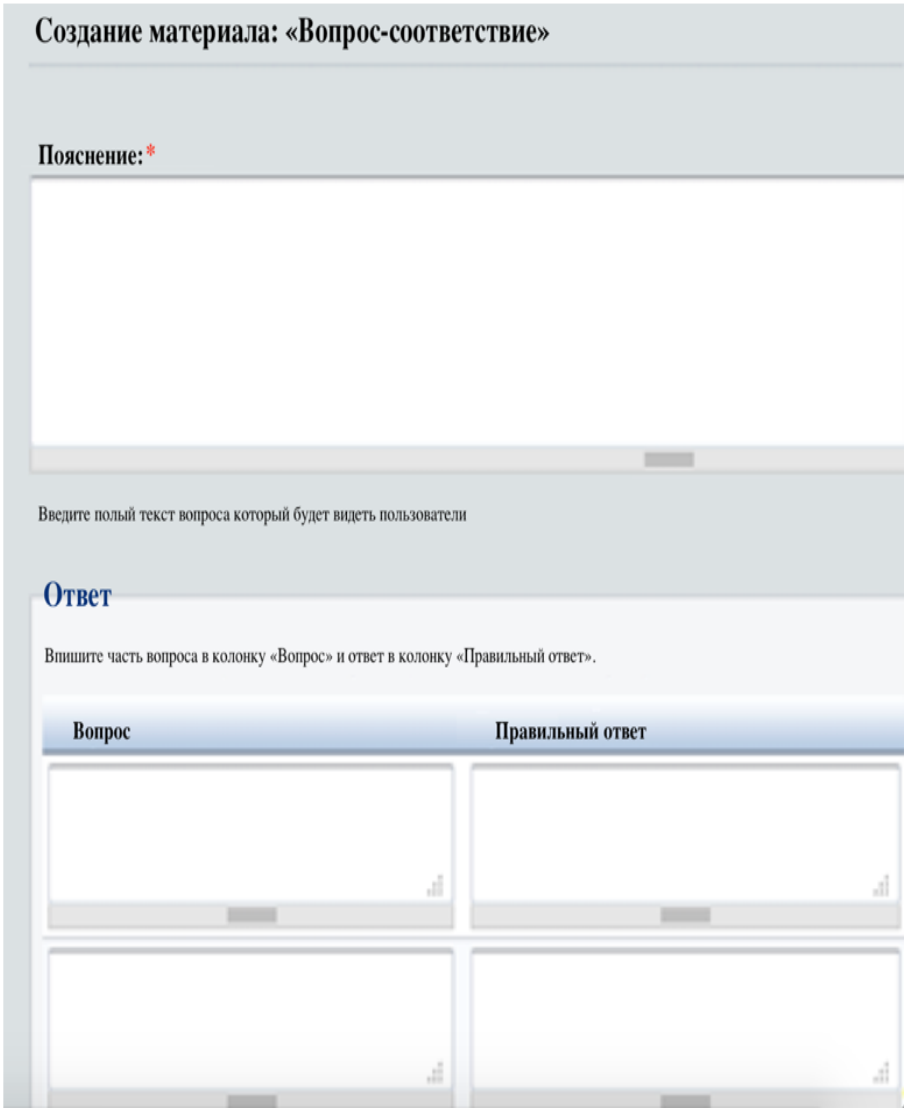


Рисунок 5.13 – Форма додавання нового питання з відповідними відповідями

У даній формі вам потрібно вказати пояснення до питання, щоб користувач зрозумів, як йому відповідати на питання. Цей тип питання, можна легко перетворити в питання-порядок, якщо замість підпитань поставити числа. Потім ви можете додати питання в уже існуючий тест або в кілька тестів, або ж створити для даного питання новий тест.

Після того, як всі питання створені, і частина з них додана в тест, ви можете додати всі питання в зацікавлений вас тест за допомогою пункту «Управління питаннями». Для цього потрібно зайти в необхідний вам тест і перейти за відповідним посиланням в форму управління питаннями (рис. 5.14).

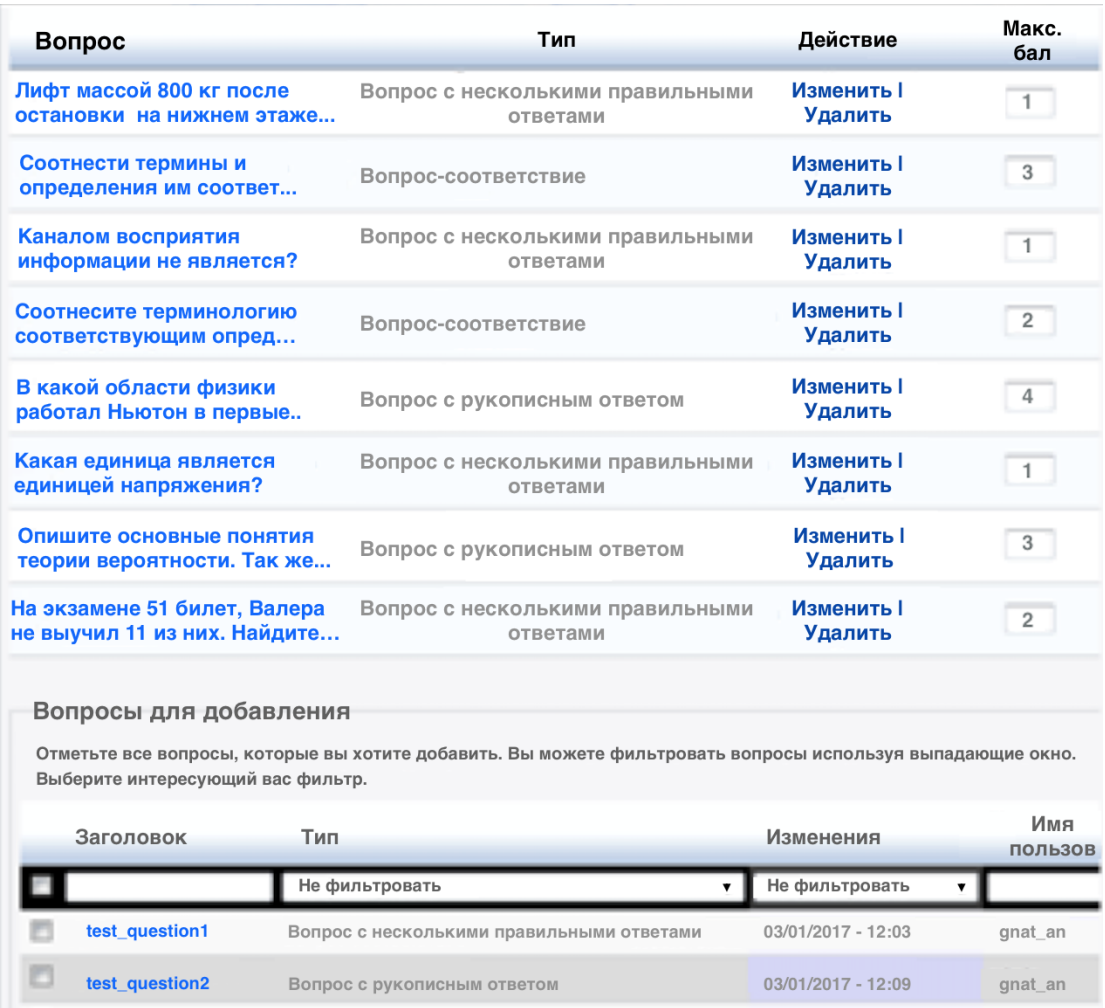


Рисунок 5.14 – Форма управління питаннями у профілі викладача

У даній формі ви можете побачити всі питання, які вже додані в тест, їх тип і максимальну кількість балів за повністю правильну відповідь, а так само список питань, які не включені ні в один тест. Потім ви можете додати ваші запитання в свій тест.

5.2.2 Проходження тестування у підсистемі планування та оцінювання знань

Для того аби користувачеві пройти тестування у системі, достатньо натиснути на головному меню «Testing» й там обрати доступні для нього тести. Існую також можливість перейти до профіля того або іншого викладача та подивитися на доступні для нього тести. Обравши тему тестування, користувач має змогу побачити питання тесту. Користувачеві необхідно вибрати тест і натиснути кнопку «Почати тестування», після чого, запуститься тестування. Процес тестування показаний на рис. 5.15.

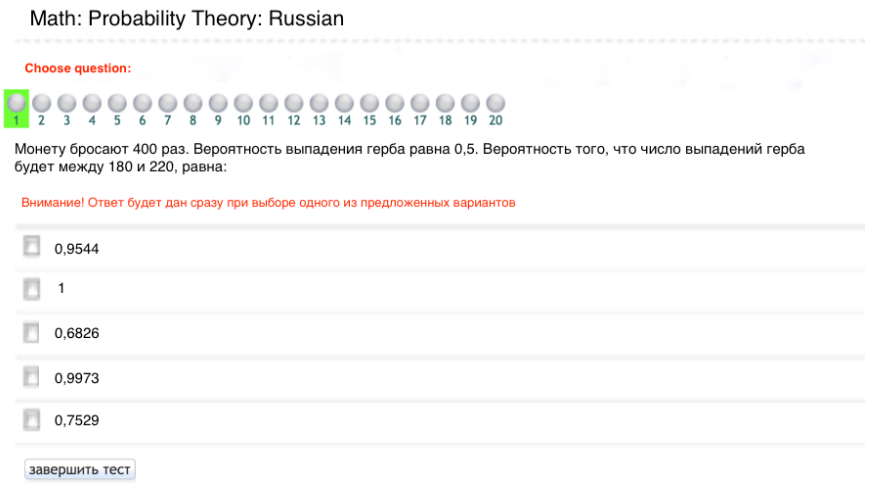


Рисунок 5.15 – Процес тестування у підсистемі планування та оцінювання знань

Після того, як користувач пройшов тестування на екрані з'явиться його результат.Викладач має можливість переглянути всі результати по всім кту-вачам і по кожному користувачеві окремо. У формі на рис. 5.16 можна врати сортування користувачів по імені, по даті початку тестуванні, за дтою здачі і по результату.

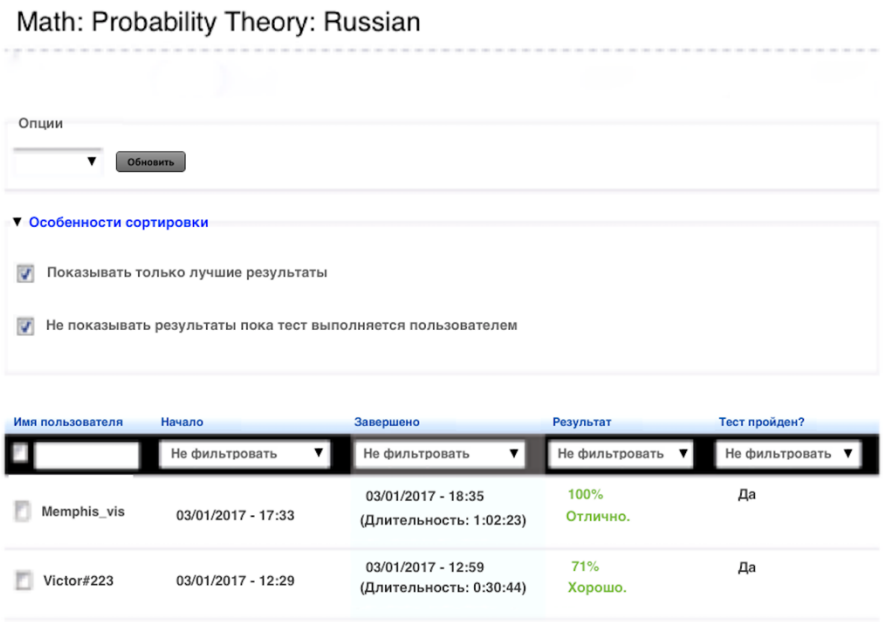


Рисунок 5.16 – Результати користувачів та їх сортування

ВИСНОВКИ

В результаті написання магістерської роботи реалізована підсистема проведення планування та оцінювання знань студентів, що включає, крім власне тестування, цикл підготовки тестів та механізми планування процесу тестування з детальним аналізом отриманих результатів студентів. В ході написання магістерської роботи потрібно було встановити і конфігурувати необхідні програмні продукти, розробити базу даних і призначений для користувача інтерфейс.

В ході написання магістерської роботи було проаналізовано безліч систем інтернет тестування, виявлено безліч плюсів і мінусів і всі вони були враховані при розробці підсистеми проведення планування та оцінювання знань.

Була реалізована зручна підсистема для тестування, також був розроблений зручний профіль користувача-викладача який може здійснити перевірку результатів після проходження тестування.

Робота написана з використанням сучасних інформаційних технологій: WEB використаний – сервер Apache, мова сценаріїв PHP, база даних MySQL. Всі ці засоби використовані в роботі.

В процесі роботи вивчена предметна область, сформульовані вимоги до системи, проведено проектування бази даних, розроблена архітектура системи і призначений для користувача інтерфейс, проведена її декомпозиція на модулі, створена програмна реалізація. Всі ці етапи так само докладно описані в магістерській роботі, програмнийкод наведений в додатку. Основним результатом проектування стало реалізація вимог, функціональних можливостей і бізнес-логіки.

ПЕРЕЛІК ПОСИЛАНЬ

1. Аванесов В. С. Композиция тестовых заданий. – М.: Центр тестирования, 2002. – 239 с.
2. Челышкова М. Б. Теория и практика конструирования педагогических тестов. Уч. Пособие. — М.: Логос, 2002. – 432 с.
3. Береза А. М. Основи створення інформаційних систем: Навчальний посібник. – К.: КНЕУ, 1998. – 140 с.
4. Хокинс С. Администрирование Web-сервера Apache и руководство по электронной коммерции – М.: Вильямс, 2001. – 336 с.
5. Видео уроки PHP MySQL БЕСПЛАТНО [Електронный ресурс] – Режим доступа: http://life-prog.ru/video.php
6. Карпова Т.С. Базы данных: модели, разработка, реализация. – СПб.: Питер, 2001. – 304 с.
7. Єрьоміна Н.В. Проектування баз даних: Навч. Посібник – К.: КНЕУ, 1998. – 208 с.
8. Крис Дж. Дейт. Введення в системибазданих. Пер. з англ. – К.: Птицин, 2006. – 1328 с.
9. Ситник В.Ф. та ін. Основи інформаційних систем: Навч. Посібник – 2-е вид., перероб. і доп. – К.: КНЕУ, 2001. – 420 с.
10. Кристофер Косентино. PHP: Web-профессионалам. К.: BHV, 2001. – 300 с.
11. Бесплатная коллекция уроков для создания сайтов [Електронный ресурс] – Режим доступа: http://ajaxs.ru/
12. Л. Аргерих, В. Чой, Д. Коггсхол и др. PHP. Профессиональное пграммирование – СПб.: Символ, 2003. – 1048 с.
13. Б. Хеник. HTML и CSS. Путь к совершенству – СПб.: Питер, 2011. – 336 с.
14. Даккет Д. HTML и CSS. Разработка и дизайн веб-сайтов. Пер. з англ. – М.: Эксмо, 2013. – 480 с.

ДОДАТОКА
ОСНОВНІКОДИПРОГРАМНИХМОДУЛІВ

Код адміністраторської частини

```
<?php defined('BASEPATH') OR exit('No direct script access al-
lowed');

class Admin extends CI_Controller {

  public function __construct(){
          parent::__construct();

          $query = $this->db->query("Select * From `admin`");
          $res = $query->result_array();
          $valid_passwords = array ();
          foreach ($res as $item) {
                  $valid_passwords [$item ['login']] = $item
['password'];
          }


          $valid_users = array_keys($valid_passwords);

          $user = $_SERVER['PHP_AUTH_USER'];
          $pass = sha1($_SERVER['PHP_AUTH_PW']);

          $validated = (in_array($user, $valid_users)) && ($pass ==
$valid_passwords[$user]);

          if (!$validated) {
            header('WWW-Authenticate: Basic realm="My Realm"');
            header('HTTP/1.0 401 Unauthorized');
            die ("Not authorized");
          }

  }

  public function index(){
          $this->users(0);
  }

  public function users($from = 0){
          $query = $this->db->query('Select COUNT(*) as `count`
From users;');
          $res  = $query->row_array();

          $this->load->library('pagination');

          $per_page = 50;

          $config['base_url'] = '/index.php/admin/users/';
```

```
            $config['total_rows'] = $res ['count'];
            $config['per_page'] = $per_page;

            $this->pagination->initialize($config);

            $data = $this->default_view_data->default_data();
            $data ['pages'] =  $this->pagination->create_links();

            $query = $this->db->query("Select * From users Limit
$from, $per_page;");
            $data ['users'] =  $query->result_array();


            $this->load->view('admin/header');
            $this->load->view('admin/users', $data);
            $this->load->view('admin/sidebar');
            $this->load->view('admin/footer');
    }

    public function user($id){


            if ( $_SERVER['REQUEST_METHOD'] == 'POST' && $_POST
['act'] == 'block'){
                    $this->db->query("UPDATE users SET `status` = 2
WHERE id = $id");
            } elseif ($_SERVER['REQUEST_METHOD'] == 'POST' && $_POST
['act'] == 'unblock'){
                    $this->db->query("UPDATE users SET `status` = 1
WHERE id = $id");
            }

            $data['action']['block'] ['button'] = 'Заблокировать';
            $data['action']['block'] ['operation'] = 'block';

            $query = $this->db->query("SELECT * FROM users WHERE id =
$id;");
            $data ['user'] = $query->row_array();

            $query = $this->db->query("SELECT * FROM tutorials WHERE
user_id = $id ORDER BY 1 DESC LIMIT 10;");
            $data ['tutorials'] = $query->result_array();

            $query = $this->db->query("SELECT * FROM questions WHERE
user_id = $id ORDER BY 1 DESC LIMIT 10;");
            $data ['questions'] = $query->result_array();

            if ( $data ['user'] ['status'] == 2) {
                    $data['action']['block'] ['button'] =
'Разблокировать';
                    $data['action']['block'] ['operation'] =
'unblock';
            }
```

```
        $this->load->view('admin/header');
        $this->load->view('admin/profile', $data);
        $this->load->view('admin/sidebar');
        $this->load->view('admin/footer');
  }


  public function tutorial($id) {


        if ( $_SERVER['REQUEST_METHOD'] == 'POST' && $_POST
['act'] == 'block'){
                $this->db->query("UPDATE tutorials SET `status` =
3 WHERE id = $id");
        } elseif ($_SERVER['REQUEST_METHOD'] == 'POST' && $_POST
['act'] == 'pub'){
                $this->db->query("UPDATE tutorials SET `status` =
1 WHERE id = $id");
        } elseif ($_SERVER['REQUEST_METHOD'] == 'POST' && $_POST
['act'] == 'cancel'){
                $this->db->query("UPDATE tutorials SET `status` =
2 WHERE id = $id");
        }

        $this->load->model('tutorial_model', 'tutorial');

        $data ['tutorial'] = $this->tutorial->get_tutorial( $id);
        $data ['files'] = $this->tutorial-
>get_files_by_tutorial($id);
        $data ['rates'] = $this->tutorial->get_rates($id);

        if ($data ['tutorial'] ['status'] == 0){
                $data ['action'] ['btn'] ['act'] = 'pub';
                $data ['action'] ['btn'] ['text'] = 'Одобрить';
                $data ['action'] ['btn2'] ['act'] = 'cancel';
                $data ['action'] ['btn2'] ['text'] = 'Отклонить';
        } elseif ($data ['tutorial'] ['status'] == 1) {
                $data ['action'] ['btn'] ['act'] = 'block';
                $data ['action'] ['btn'] ['text'] =
'Заблокировать';
        } elseif ($data ['tutorial'] ['status'] == 2 || $data
['tutorial'] ['status'] == 3) {
                $data ['action'] ['btn'] ['act'] = 'pub';
                $data ['action'] ['btn'] ['text'] = 'Опубликовать';
        }


        $this->load->view('admin/header');
        $this->load->view('admin/tutorial', $data);
        $this->load->view('admin/sidebar');
        $this->load->view('admin/footer');
  }
```

```
  public function question($id) {


          if ( $_SERVER['REQUEST_METHOD'] == 'POST' && $_POST
['act'] == 'block'){
                  $this->db->query("UPDATE questions SET `status` =
2 WHERE id = $id");
          } elseif ($_SERVER['REQUEST_METHOD'] == 'POST' && $_POST
['act'] == 'pub'){
                  $this->db->query("UPDATE questions SET `status` =
0 WHERE id = $id");
          }

          $this->load->model('question_model', 'question');

          $data ['question'] = $this->question->get_question( $id);
          $data ['files'] = $this->question-
>get_files_by_question($id);

          if ($data ['question'] ['status'] == 0) {
                  $data ['action'] ['btn'] ['act'] = 'block';
                  $data ['action'] ['btn'] ['text'] =
'Заблокировать';
          } elseif ($data ['question'] ['status'] == 2) {
                  $data ['action'] ['btn'] ['act'] = 'pub';
                  $data ['action'] ['btn'] ['text'] = 'Одобрить';
          }


          $this->load->view('admin/header');
          $this->load->view('admin/question', $data);
          $this->load->view('admin/sidebar');
          $this->load->view('admin/footer');
  }



  public function loadfile($sys_name){

          $query = $this->db->query("SELECT * FROM `files` WHERE
`sys_name` = '$sys_name'");
          $res = $query -> row_array();

          $file_url = 'uploads/' . $sys_name;
          header('Content-Type: application/octet-stream');
          header("Content-Transfer-Encoding: Binary");
          header("Content-disposition: attachment; filename=\""
.$res ['file_name'] . "\"");
          readfile($file_url);
  }

  public function search(){
```

```
        $data ['objects'] = array();

        if ($_SERVER ['REQUEST_METHOD'] == 'POST' && isset($_POST
['search-text']) && isset( $_POST['type'] )) {

                switch ( $_POST['type']) {
                        case '1':
                        $table = 'questions';
                                break;
                        default:
                        $table = 'tutorials';
                                break;
                }

                $this->db->select('t.id, t.title, t.description,
t.price ,FROM_UNIXTIME(t.date) as date, c.name as categoria');
                $this->db->from($table.' t, categorias c');
                $this->db->where("t.categoria = c.id AND
t.`status` = 1 AND `title` LIKE '%{$_POST ['search-text']}%' OR
`description` LIKE '%{$_POST ['search-text']}%'");
                $this->db->order_by('t.date', 'desc');
                $this->db->group_by('t.id');

                $data ['objects'] = $this->db->get() -> re-
sult_array();
        }


        $this->load->view('admin/header');
        $this->load->view('admin/search', $data);
        $this->load->view('admin/sidebar');
        $this->load->view('admin/footer');
  }


  public function tutorials(){

        $data = array();

        $this->db->select('t.id, t.title, t.description, t.price
,FROM_UNIXTIME(t.date) as date, c.name as categoria');
        $this->db->from('tutorials t, categorias c');
        $this->db->where('t.categoria = c.id');
        $this->db->where("t.status = 0");
        $this->db->order_by('t.date', 'desc');

        $data ['tutorials'] = $this->db->get() -> result_array();

        $this->load->view('admin/header');
        $this->load->view('admin/tutorials', $data);
        $this->load->view('admin/sidebar');
        $this->load->view('admin/footer');
```

```
  }

  public function questions(){

          $data = array();

          $this->db->select('t.id, t.title, t.description, t.price
,FROM_UNIXTIME(t.date) as date, c.name as categoria');
          $this->db->from('questions t, categorias c');
          $this->db->where('t.categoria = c.id');
          $this->db->where("t.status = 0");
          $this->db->order_by('t.date', 'desc');

          $data ['questions'] = $this->db->get() -> result_array();

          $this->load->view('admin/header');
          $this->load->view('admin/questions', $data);
          $this->load->view('admin/sidebar');
          $this->load->view('admin/footer');
  }

  public function comments(){


          if ($_SERVER ['REQUEST_METHOD'] == "POST") {
                  $this->db->query("UPDATE comments SET status =
{$_POST ['act']} WHERE id = {$_POST ['id']}");
          }

          $query = $this->db->query('select
                  users.id as author_id,
                  CONCAT(users.firstname, \' \', users.lastname) as
author,
                  comments.id,
                  comments.comment,
                  comments.date,
                  tutorials.id as t_id,
                  tutorials.title as t_title
          from
                  comments,
                  users,
                  tutorials
          where
                  comments.user_id = users.id AND
                  comments.tutorial_id = tutorials.id AND
                  comments.`status` = 0;');


          $data ['comments'] = $query->result_array();

          $this->load->view('admin/header');
          $this->load->view('admin/comments', $data);
          $this->load->view('admin/sidebar');
```

```
        $this->load->view('admin/footer');
  }

  public function withdraw() {

        if ($_SERVER ['REQUEST_METHOD'] == "POST") {
                $this->db->query("UPDATE actions SET status =
{$_POST ['act']} WHERE id = {$_POST ['id']}");
                $this->db->query("Call up-
date_balance({$_POST['user']})");
        }

        $data = array();

        $query = $this->db->query("select
                actions.id,
                actions.system_cost as price,
                actions.`comment`,
                users.id as author_id,
                CONCAT(users.firstname, ' ', users.lastname) as
author,
                actions.date_created
        from
                actions,
                users
        where
                actions.user_id = users.id AND
                users.`status` = 1 AND
                actions.pay_system = 0 AND
                actions.`status` = 0
        ORDER BY actions.date_created ASC;");

        $data ['withdraw'] = $query->result_array();


        $this->load->view('admin/header');
        $this->load->view('admin/widthraw', $data);
        $this->load->view('admin/sidebar');
        $this->load->view('admin/footer');
  }
}
```

Клас для роботи з користувачами

```
<?php defined('BASEPATH') OR exit('No direct script access al-
lowed');

class User extends CI_Model {

   public function exists_email($email) {

  $this->db->select('COUNT(*) as `count`');
  $this->db->from('users');
```

```
  $this->db->where('email', $_POST['email']);

  return $this->db->get() -> row_array() ['count'];
   }

  public function add_new_user($email, $password, $firstname,
$lastname){

          $this->load->helper('password');
          $time = time();

          $this->db->insert('users', array(
                                          "email" => $email,
                                          "lastname" => $lastname,
                                          "firstname" =>
$firstname,
                                          "password" =>
get_hash_password($email, $password, $time),
                                          "date_register" => $time,
                                          "mail_link" =>
sha1($email . $time . 'Make-HomeWork'),
                                          "email_message" => 1,
                                          "status" => 1
          ));

  }

  public function auth($email, $password){

                  $this->load->helper('password');

                  $this->db->select('id, password, date_register');
                  $this->db->from('users');
                  $this->db->where("email = '{$_POST['email']}' AND
`status` in (0,1)");
                  $result = $this->db->get() -> row_array();
                  if ( ! $result ) {
                          return false;
                  } else if ( $result ['password'] ==
get_hash_password($_POST ['email'], @$_POST ['password'], $result
['date_register']) ) {
                          add_session_field('user_id', $result
['id']);
                          return true;
                  } else {
                          return false;
                  }

  }


  public function set_new_password($user_id, $password,
$new_password){
```

```
                $this->load->helper('password');

                $this->db->select('email, password,
date_register');
                $this->db->from('users');
                $this->db->where('id', $user_id);

                $result = $this->db->get() -> row_array();

                if ( ! $result ) {
                        return false;
                } else if ( $result ['password'] ==
get_hash_password($result ['email'], $password, $result
['date_register']) ) {

                        $password = get_hash_password($result
['email'], $new_password, $result ['date_register']);
                        $this->db->query("UPDATE users SET `pass-
word` = '$password' WHERE `id` = $user_id;");

                        return true;
                } else {
                        return false;
                }

  }


  public function get_count_hisotry_balance($user_id){
          $query = $this->db->query('SELECT COUNT(*) as `count`
FROM actions WHERE user_id = '. $user_id);
          return $query -> row_array() ['count'];
  }

  public function get_history_balance($user_id,$from, $limit){
          $query = $this->db->query('SELECT
FROM_UNIXTIME(actions.date_created) as `date`, actions.status, ac-
tions.system_cost,
          tutorial_id as `t_id`, tutorials.title as `t_title`,
          question_id as `q_id`, questions.title as `q_title`
                  FROM actions LEFT JOIN tutorials ON ac-
tions.tutorial_id = tutorials.id
                                  LEFT JOIN questions ON ac-
tions.question_id = questions.id  WHERE actions.user_id =\'. $us-
er_id  ." ORDER BY actions.date_created DESC LIMIT $from, $limit;
");

          return $query -> result_array();
  }
}
```

Функция добавления пользователя в БД

```
<?php
function insertUser()
{
if( isset($_SESSION['login']) && isset($_SESSION['password']) &&
isset($_SESSION['passwordagain']) && isset($_SESSION[email]) )
{
if($_SESSION['password'] == $_SESSION['passwordagain'])
{
$res = mysql_query("SELECT * FROM User_List WHERE Login =
'$_SESSION[login]' ; ")or die("ERROR ".mysql_error());
$num = mysql_num_rows($res);
if($num == '0')
{
$s = strftime("%S")+1;
$m = strftime("%M")+1;
$h = strftime("%H")+1;
$d = strftime("%d")+1;
$mm =strftime("%m")+1;
$id = $s * $m * $h *$d *$mm;
$res = mysql_query("INSERT INTO User(UID,Login,Pass,Email,init)
VAL-
UES('$id','$_SESSION[login]','$_SESSION[password]','$_SESSION[emai
l]','0'); ")or die("ERROR ".mysql_error());
if(res)
{
//$_SESSION['message'] = "Регистрация прошла успешно.";
include_once("auth.php");
additionalData();
verification();
}
echo"$res";
}
else
$_SESSION['message'] = "Данный логин уже зарегистрирован в
системе.";
}
else
$_SESSION['message'] = "Несовпадение паролей.";
}
else
$_SESSION['message'] = "Заполните все поля.";
}
?>
```

Проверка данных пользователя

```
<?php
function verification()
{
include_once("db.php");
if( isset($_SESSION['login']) && isset($_SESSION['password']) &&
!empty($_SESSION['login']) && !empty($_SESSION['password']) )
```

```
{
$res = mysql_query("SELECT * FROM User WHERE Login =
'$_SESSION[login]' AND Pass = '$_SESSION[password]' ; ")or
die("ERROR ".mysql_error());
$num = mysql_num_rows($res);
if($num == '0')
{
$res = mysql_query("SELECT * FROM User WHERE Login =
'$_SESSION[login]' ; ")or die("ERROR ".mysql_error());
$num = mysql_num_rows($res);
if( $num == '0' )
$_SESSION['message'] = "Данный логин не зарегистрировани в
системе.";
else
$_SESSION['message'] = "Неверная комбинация лигина и пароля.";
}
else
{
$tmp = mysql_fetch_array($res);
$_SESSION['UID'] = $tmp['UID'];
if( !get_magic_quotes_gpc() )
{
$_SESSION['login'] = mysql_escape_string($_SESSION['login']);
$_SESSION['pass'] = mysql_escape_string($_SESSION['pass']);
}
$_SESSION['login'] = $_SESSION['login'];
$_SESSION['pass'] = $_SESSION['pass'];
unset($_GET['act']);
unset($_SESSION['login']);
unset($_SESSION['pass']);
echo "<HTML><HEAD><META HTTP-EQUIV='Refresh' CONTENT='0;
URL=index.php?act=groups'></HEAD></HTML>";
}
}
else
$_SESSION['message'] = "Пожалуйста, заполните поля: Логин и
Пароль.";
}?>
```

Функция вывода на экран полей регистрации

```
<?php
function registration()
{?>
<div id="content">
<div id="posts">
<div class="post">
<h2 class="title">Регистрация : </h2>
<form method = post>
<table>
<tr>
<td><p>Логин:</p></td>
<td><input type=text name=login value=''><br></td>
```

```
</tr>
<tr>
<td><p>Пароль:</p></td>
<td><input type=password name=password value=''><br></td>
</tr>
<tr>
<td><p>Пароль повторно:</p></td>
<td><input type=password name=passwordagain value=''><br></td>
</tr>
<tr>
<td><p>E-mail:</p></td>
<td><input type=text name=email value=''><br></td>
</tr>
<tr>
<td></td>
<td><input type = submit value = 'Регистрация' ></td>
</tr>
</table>
</form>
<?php
if(isset($_SESSION['message']))
{?>
<div id="content">
<div id="posts">
<div class="post">
<div class="story">Сообщение :
<?php
echo" $_SESSION[message] ";
unset($_SESSION['message']);
?>
</div>
</div>
</div>
</div>
<?php
}?>
</div>
</div>
</div>
<?php }?>
```