**Міністерство освіти і науки України**

**Чернівецький національний університет**

**імені Юрія Федьковича**

Факультет математики та інформатики

(повна назва інституту/факультету)

Кафедра прикладної математики

(повна назва кафедри)

**Курсова робота**

**на тему: «Реалізація веб-сайту виставлення оцінок з використанням**

**технології BlockChain»**

Студента 4 курсу 402 групи

Спеціальності 113 – Прикладна математика

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(прізвище та ініціали)

Керівник \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Національна шкала \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кількість балів: \_\_\_\_Оцінка: ECTS \_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис керівника) (прізвище та ініціали)

Чернівці – 2019

**Анотація**

У курсовій роботі розроблена веб-сайт в візуальному середовищі Visual Studio 2019, в якій реалізовано технологія BlockChain.

**Зміст**

[**Вступ** 3](#_Toc35711647)

[**Розділ I. Використані технології** 4](#_Toc35711648)

[**1.1. Блок транзакцій** 4](#_Toc35711649)

[**1.2. Ланцюжок блоків** 6](#_Toc35711650)

[**1.3. BlockChain** 8](#_Toc35711651)

[**Розділ 2. Опис програмного продукту** 8](#_Toc35711652)

[**2.1 Інструкція по використанню** 8](#_Toc35711653)

[**2.2 Реалізація алгоритму** 15](#_Toc35711654)

[**Висновки** 18](#_Toc35711655)

[**Перелік посилань** 19](#_Toc35711656)

[**Додаток** 20](#_Toc35711657)

**Вступ**

Для збереження даних і їх захисту від спотворення використаєм технологію BlockChain.

В перше цей метод було описано Стюартом Хабером та У. Сккотом Сторнеттою.

Робота написана на основі першого розділу, в якій розглянуто технологія BlockChain, ми програмно реалізовуємо цю технологію для збереження інформації про оцінки студентів.

Робота складається з двох розділів та додатку, в якому надається програмний код.

Перший розділ – використані технології. В ній розглянемо принцип технології BlockChain.

Другий розділ – опис програмного продукту, який складається з двох частин:

1. Перша частина.

В ній ми надаєм загальні рекомендацій до використання програмного продукту.

1. Друга частина.

В цій частині буде описано програмну реалізацію методу.

**Розділ I. Використані технології**

* 1. **Блок транзакцій**

Блок транзакцій — спеціальна структура для запису нових транзакцій в системі Біткоїн та аналогічних їй.

Блок містить відомості про транзакції, дерево їхніх гешів, а також заголовок зі службовими даними, де зокрема наведено і геш попереднього блока, тож кожен наступний блок є також підтвердженням попереднього.

Щоб транзакція вважалася достовірною («підтвердженою»), її формат та підписи мусять перевірити й записати (разом з іншими транзакціями) в новий блок. Але справді надійна перевірка достовірності транзакції потребує наявності декількох наступних блоків. Кожен наступний блок посилається на попередній, тож усі блоки можна вишикувати в один ланцюжок, що являтиме собою історію транзакцій за ввесь час існування системи. Перший блок ланцюжка — первинний блок (англ. genesis block) — то окремий випадок, бо в нього відсутній материнський блок.

Блок складається із заголовка та списку транзакцій. Заголовок блоку містить свій геш, геш попереднього блоку, геші транзакцій та додаткову службову інформацію. Першою транзакцією в блоці завжди вказується отримання комісії, яка стане нагородою користувачеві за створений блок.

Далі йдуть всі або деякі з останніх транзакцій, які ще не були записані в попередні блоки. Для транзакцій в блоці використовується деревисте гешування, аналогічне формуванню геш-суми файлу в протоколі BitTorrent. Транзакції, крім нарахування комісії за створення блоку, містять всередині атрибута input посилання на транзакцію, за якою на цей рахунок були отримані біткоїни. Комісійні операції можуть містити в атрибуті будь-яку інформацію (для них це поле носить назву англ. Coinbase parameter), оскільки у них немає батьківських транзакцій.

Створений блок буде прийнятий іншими користувачами, якщо числове значення гешу заголовка менше або дорівнює певному числу, величина якого періодично коригується. Оскільки результат гешування (функції SHA-256) необоротний, немає алгоритму отримання бажаного результату, окрім повного перебору чи пошуку навмання. Якщо геш не задовольняє умову, то довільно змінюється блок службової інформації в заголовку, а геш обчислюється знов. Зазвичай потрібно чимало переобчислень. Коли умову дотримано, вузол висилає створений блок іншим підключеним вузлам, а ті його перевіряють. Якщо помилок немає, то блок вважається доданим в ланцюжок, і вже його геш міститиме наступний блок.

Величина цільового числа, з яким порівнюється геш, коригується через кожні 2016 блоків. Заплановано, що вся мережа витрачатиме на створення одного блоку приблизно 10 хвилин, на 2016 блоків — близько двох тижнів. Якщо 2016 блоків сформовано швидше, то ціль трохи зменшують і досягти її стає важче, інакше ціль збільшують. Зміна складності обчислень не впливає на надійність мережі Біткоїн і потрібна лише для того, щоб система створювала блоки з майже постійною швидкістю незалежно від потужності мережі.

* 1. **Ланцюжок блоків**

Над створенням нових блоків одночасно працює чимало «майнерів». Новостворений блок, що відповідає певним умовам, негайно надсилається решті членів мережі і має стати наступною ланкою ланцюжка. Постійно трапляється таке, що з різних частин мережі (від різних учасників) надходять блоки, що попереднім називають той самий блок, тобто відбувається галуження. Навмисне чи ненароком можна обмежити поширення новостворених блоків (наприклад, одне з галужень ланцюжка може деякий час розвиватися в межах локальної мережі). Тоді одночасно відбувається створення кількох гілок одного ланцюжка, що суперечать одна одній.

Коли поширення блоків поновлюється, майнери розв'язують суперечність, обираючи найдовшу гілку з найбільшим рівнем складності за єдину «достовірну». За однакової складності і довжини перевага віддається гілці, кінцевий блок якої з'явився раніше. Суперечні гілки можуть містити різні множини транзакцій, тобто не всяка транзакція конче присутня в усіх гілках. Тож транзакції, що входять лише до відхиленої гілки (зокрема, транзакції з виплати винагороди), втрачають підтвердженість.

Кожну транзакцію переказу коштів, що містилась лише у відхилених гілках, знов буде поставлено в чергу, а відтак включено в черговий блок. Натомість транзакції з одержання винагороди за створення відхилених зрештою блоків не отримають дальших підтверджень і відповідні «зайві» кошти буде втрачено.

Розподілена база даних Blockchain — це ланцюжок блоків, що постійно зростає, зберігаючи всю історію транзакцій. Копія бази даних або її частини одночасно зберігаються на безлічі комп'ютерів та синхронізуються відповідно до формальних правил побудови ланцюжка блоків. Дані блоків не шифровані і доступні у відкритому вигляді, проте захищені від змін криптографічно через геш-ланцюжок.

Зазвичай умисна зміна інформації в будь-якій копії бази або навіть в багатьох копіях не буде визнана істинною, бо не відповідатиме правилам. Деякі зміни може бути прийнято, якщо їх внести в усі копії бази (наприклад, видалення кількох останніх блоків через помилку в їхньому формуванні).

До версії 0.8.0 для зберігання ланцюжка блоків основний клієнт використовував Berkeley DB, починаючи з версії 0.8.0 розробники перейшли на LevelDB.

* 1. **BlockChain**

BlockChain - розподілена база даних, що зберігає впорядкований ланцюжок записів (так званих блоків), що постійно довшає. Дані захищено від підробки та спотворення. Кожен блок містить часову позначку, геш попереднього блока та дані транзакцій, подані як геш-дерево.

**Розділ 2. Опис програмного продукту**

**2.1 Інструкція по використанню**

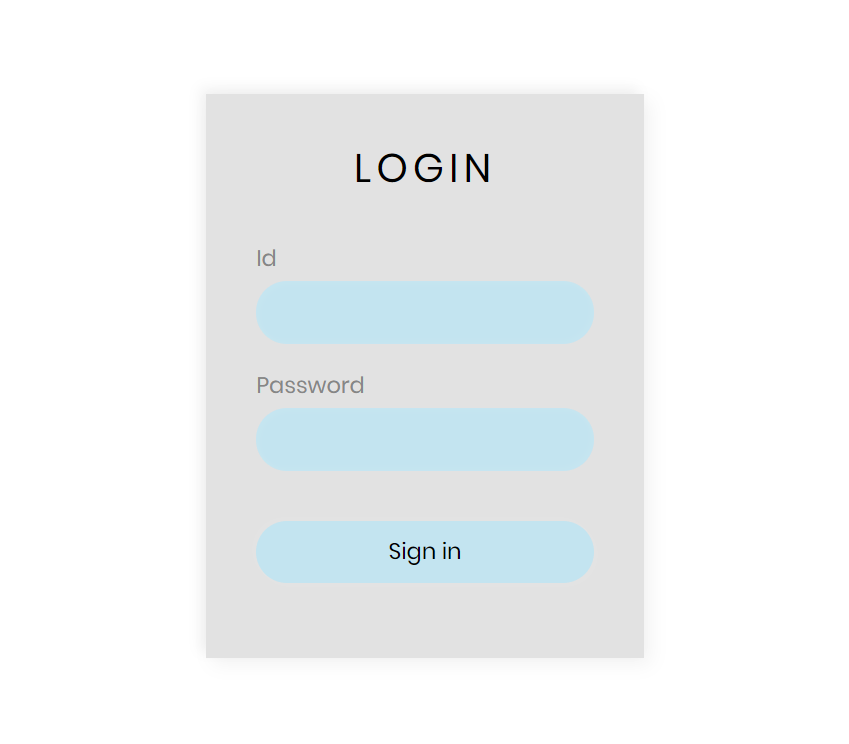
Web site містить три типи сторінок:

А) Admin

B) Student

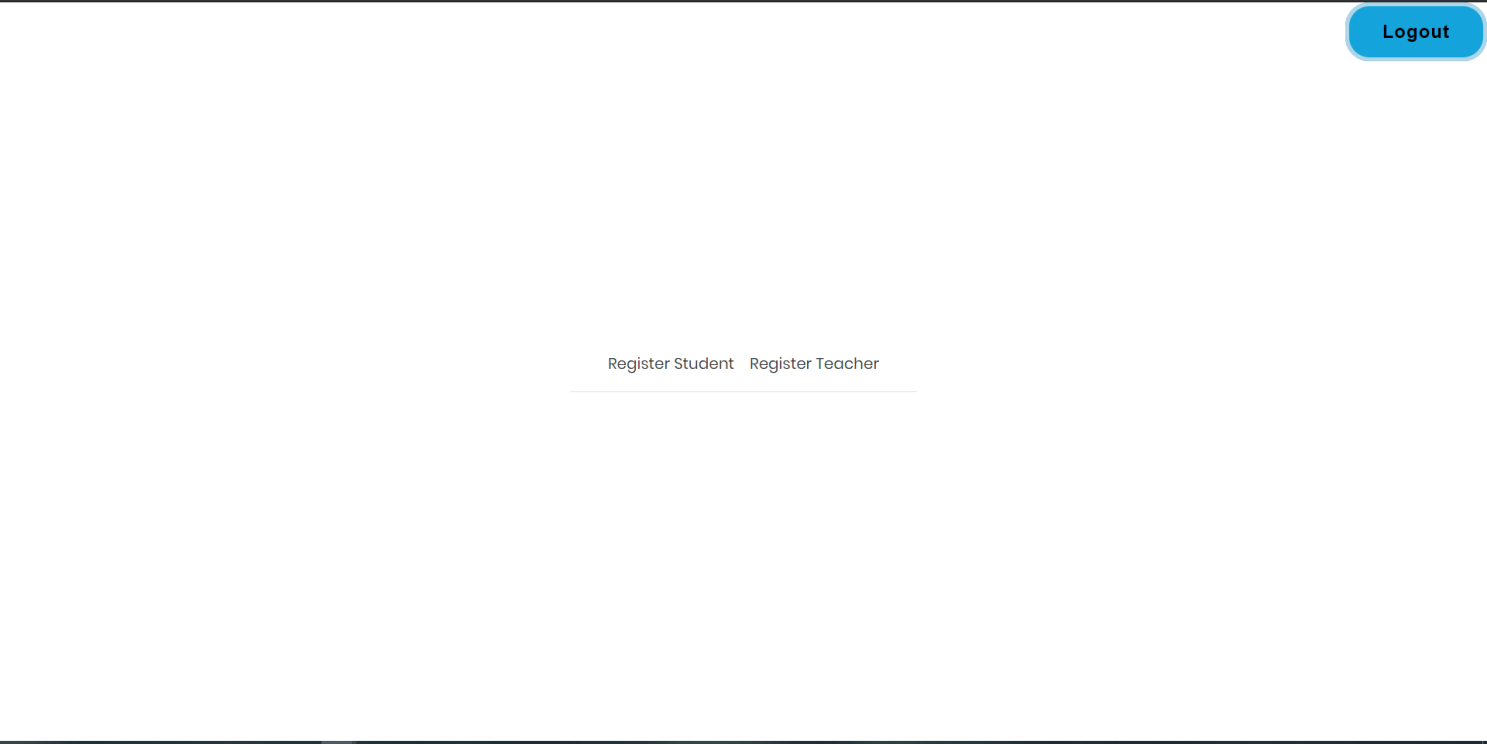
C) Teacher

При першому запуск буде відображена Login form:



**Рис.2.1**

Admin

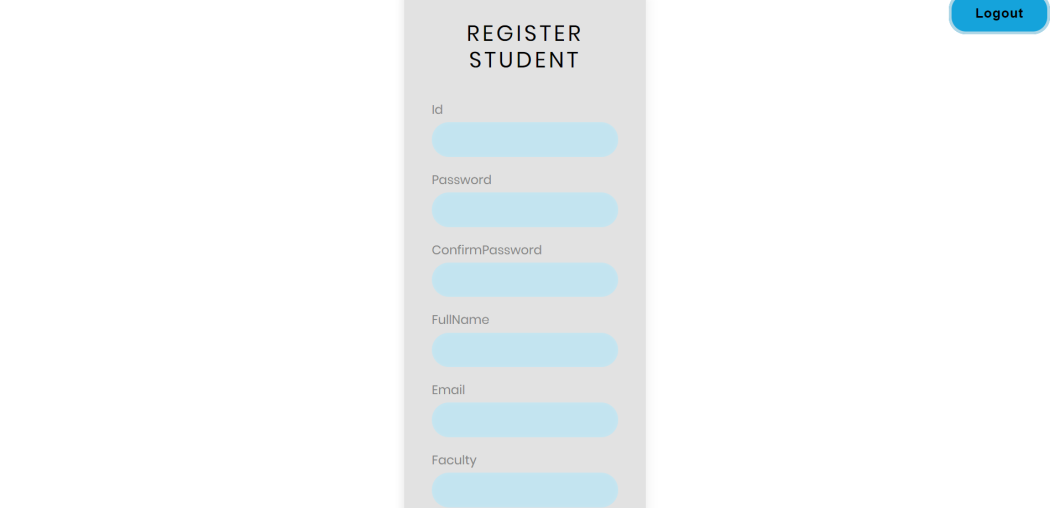


**Рис.2.2**

Index сторінка містить три кнопки:

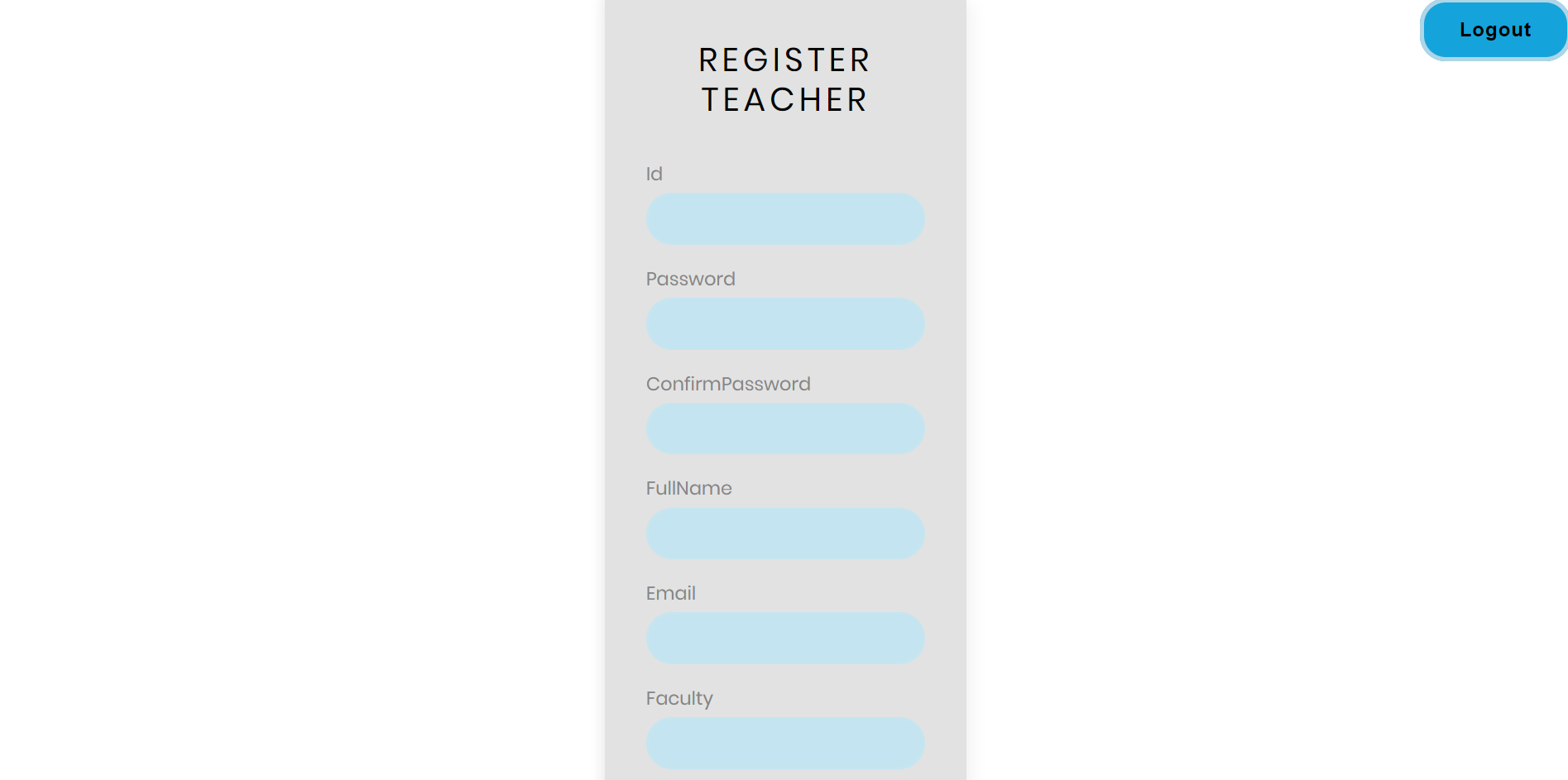
1. Register Student – перехід на форму реєстрацій студента
2. Register Teacher – перехід на форму реєстрацій викладача
3. Logout – вихід

Register student form



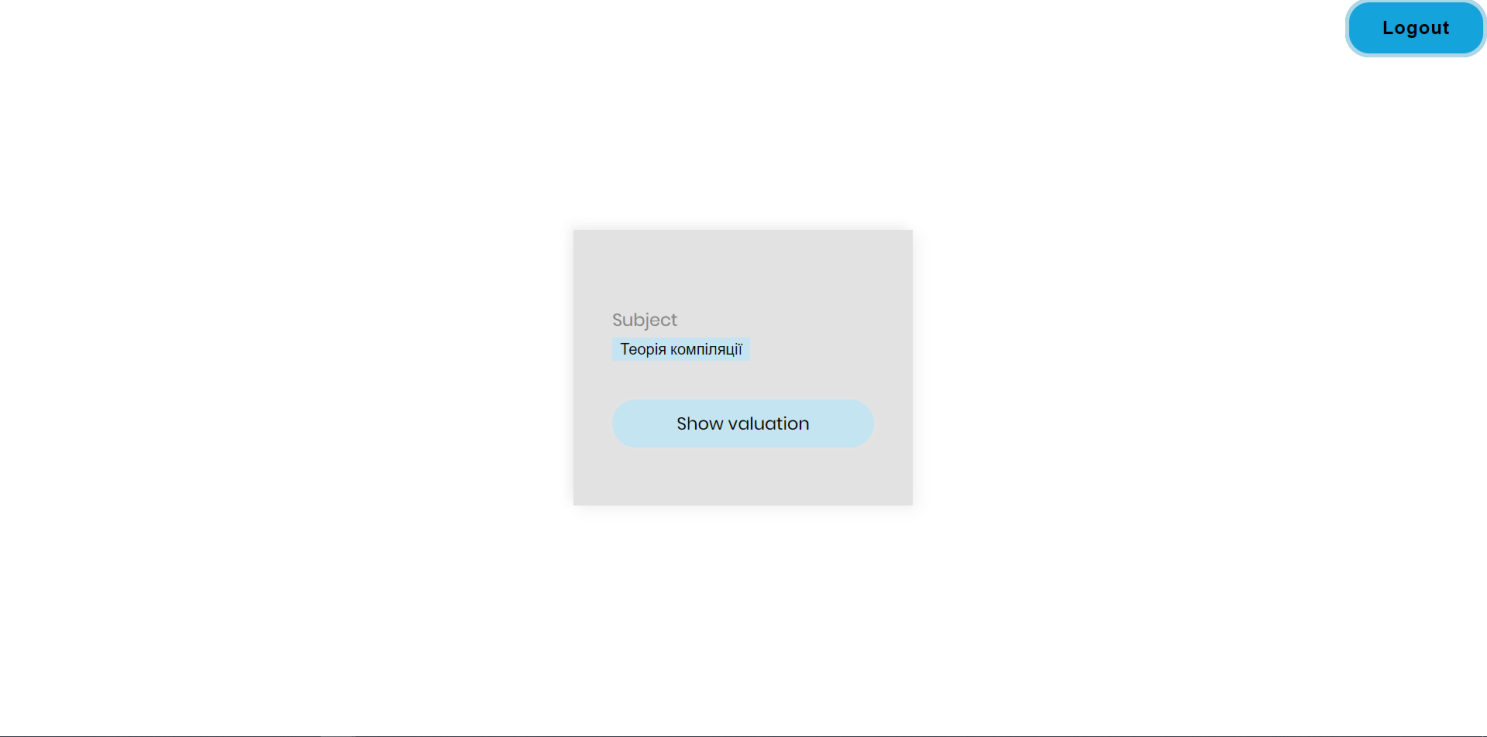
**Рис.2.3**

Register teacher form

****

**Рис.2.4**

Student

****

**Рис.2.5**

Index містить форму, в якій студент можу вибрати предмет який він вивчає і переглянути отримані оцінки і також кнопку “Logout”.

Show



**Рис.2.6**

На цій сторінці можна побачити структуру BlockChain.

Teacher



**Рис.2.7**

Index містить форму для вибору предмету та групи для виставлення і показу оцінок, а також кнопку “Logout”.

Set valuation



**Рис.2.8**

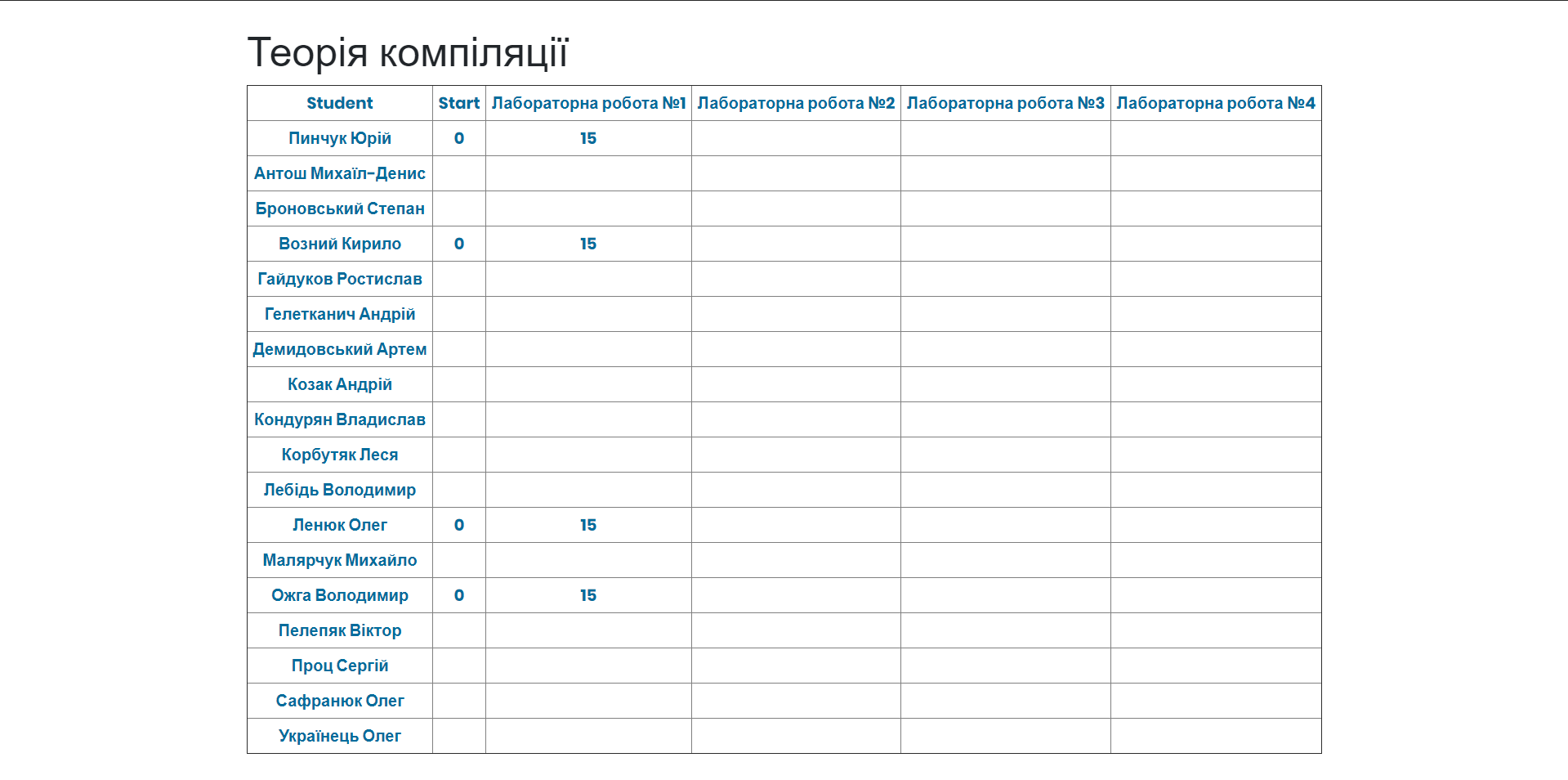
На цій сторінці можна вибрати одного чи двох чи більше студентів для виставлення оцінки.

Поле “Description” міститиме коротку пояснення виставлення оцінки.

“Amout” міститиме оцінку.

В продукті передбачено контроль доступу до сторінок. Якщо користувач спробує перейти на сторінку не дозволену йому відбудеться редірект на його Index page або Login form якщо користувач не був авторизованим.

Show valuation



**Рис.2.9**

На цій сторінці відображаються з вибраного предмету та групи.

**2.2 Реалізація алгоритму**

Для реалізації технології BlockChain було написано три класи:

1. Block
2. BlockChain
3. Valuation

Block код:

public class Block {

#region Variables

public int Index { get; set; }

public DateTime TimeStamp { get; set; }

public string PreviousHash { get; set; }

public string Hash { get; set; }

public Valuation Valuation { get; set; }

#endregion

public Block ( DateTime timeStamp, string previousHash, Valuation valuation ) {

Index = 0;

TimeStamp = timeStamp;

PreviousHash = previousHash;

Valuation = valuation;

}

public string CalculateHash () {

SHA256 sha256 = SHA256.Create();

byte[] inputBytes = Encoding.ASCII.GetBytes($"{TimeStamp}-{PreviousHash ?? ""}-{JsonConvert.SerializeObject(Valuation)}");

byte[] outputBytes = sha256.ComputeHash(inputBytes);

return Convert.ToBase64String(outputBytes);

}

}

BlockChain код:

public class BlockChain {

#region Variables

[BsonId]

public int Id { get; set; }

public string Subject { get; set; }

public List<Block> Chain { set; get; }

#endregion

public BlockChain ( string subject ) {

InitializeChain(subject);

}

public void InitializeChain ( string subject ) {

Subject = subject;

Chain = new List<Block>();

AddGenesisBlock();

}

public Block CreateGenesisBlock () {

Teacher teacher = new Teacher("","",null,"","","");

Valuation valuation = new Valuation(teacher, 0,"");

Block block = new Block(DateTime.Now, null, valuation);

return block;

}

public void AddGenesisBlock () {

Chain.Add(CreateGenesisBlock());

}

public Block GetLatestBlock () {

return Chain[Chain.Count - 1];

}

public void AddBlock ( Block block ) {

Block latestBlock = GetLatestBlock();

block.Index = latestBlock.Index + 1;

block.PreviousHash = latestBlock.Hash;

block.Hash = block.CalculateHash();

Chain.Add(block);

}

public bool IsValid () {

for(int i = 1; i < Chain.Count; i++) {

Block currentBlock = Chain[i];

Block previousBlock = Chain[i - 1];

if(currentBlock.Hash != currentBlock.CalculateHash()) {

return false;

}

if(currentBlock.PreviousHash != previousBlock.Hash) {

return false;

}

}

return true;

}

public float GetValuation () {

float valuation = 0;

foreach(Block block in Chain) {

valuation += block.Valuation.Amount;

}

return valuation;

}

}

Valuation код:

public class Valuation {

#region Variables

public Teacher Teacher { get; set; }

public float Amount { get; set; }

public string Description { get; set; }

#endregion

public Valuation ( Teacher teacher, float amount, string description ) {

Teacher = teacher;

Amount = amount;

Description = description;

}

Під час реєстрації студента створюється порожній список BlockChain.

При виставлення оцінок викладачем відбувається перевірка чи є створений BlockChain для цього предмету, якщо ні то створюється новий BlockChain і при створення генерується Genesis block.

Вставка нового блока в вже існуючий BlockChain:

1. Береться Hash посліднього блока і присвоюється PreviousHash цього блока
2. Генерується SHA256 на основі даних блока і присвоюється в Hash.
3. Блок додається в список блоків BlockChain.

**Висновки**

У курсовій роботі розроблена програма, де реалізовано BlockChain. Програма дозволяє зберігати дані за допомогою ланцюжків блоків.

Даний проект дозволяє виставляти оцінки і також побачити структуру технології BlockChain.

**Перелік посилань**

1. <https://uk.wikipedia.org/wiki/BlockChain> -BlockChain
2. <https://blockgeeks.com/guides/what-is-blockchain-technology/> - BlockChain
3. <https://docs.mongodb.com/> - Mongodb documentation
4. <https://docs.microsoft.com/en-us/aspnet/core/?view=aspnetcore-3.1> –ASP.NET CORE documentation
5. <https://docs.microsoft.com/en-us/aspnet/mvc/> - ASP.NET MVC documentation
6. <https://github.com/YuraPunchyk/BlockChainWeb> - project code on github

**Додаток**