

Міністерство освіти та науки України  
Одеський національний політехнічний університет  
Інститут комп'ютерних систем  
Кафедра інформаційних систем

## **КУРСОВА РОБОТА**

з дисципліни “Технології створення програмного продукту”  
за темою “Сайт-сервіс для навчання – Teachable moment”

Виконала:  
студентка 3-го курсу  
групи АІ-183  
Гірля А. І.

Перевірив:  
Блажко О. А

Одеса - 2020

## Анотація

В курсовій роботі розглядається процес створення програмного продукту «Сайт-сервіс для навчання – Teachable moment». Робота виконувалась в команді з декількох учасників: Ткаченко К.І., Гірля А.І., Кеосак А.І., Залуковський Ю.О. Тому в пояснювальній записці у розділах «Проектування» та «Конструювання» детальніше описано лише одну частину з урахуванням планів проведених робіт з розділу «Планування» з описом особливостей конструювання:

- структур даних моделі «JSON» в системі керування базами даних «MongoDB»;
- програмних модулів в інструментальному середовищі «WebStorm» з використанням фреймворків: «Vue.js», «Node.js», «ExpressJS» та мови програмування «Java Script».

Результати роботи розміщено на github-репозиторії за адресою:  
<https://github.com/tenn5/teachable-moment>.

## **Перелік скорочень**

ОС – операційна система

ІС – інформаційна система

БД – база даних

СКБД – система керування базами даних

ПЗ – програмне забезпечення

ПП– програмний продукт

UML – уніфікована мова моделювання

## ЗМІСТ

1	Вимоги до програмного продукту.....	6
1.1	Визначення потреб споживача.....	6
1.1.1	Ієрархія потреб споживача.....	6
1.1.2	Деталізація матеріальної потреби.....	6
1.2	Бізнес-вимоги до програмного продукту.....	7
1.2.1	Опис проблеми споживача.....	7
1.2.1.1	Концептуальний опис проблеми споживача.....	7
1.2.1.2	Метричний опис проблеми споживача.....	7
1.2.2	Мета створення програмного продукту.....	8
1.2.2.1	Проблемний аналіз існуючих програмних продуктів.....	8
1.2.2.2	Мета створення програмного продукту.....	9
1.2.3	Назва програмного продукту.....	9
1.2.3.1	Гасло програмного продукту.....	9
1.2.3.2	Логотип програмного продукту.....	10
1.3	Вимоги користувача до програмного продукту.....	10
1.3.1	Історія користувача програмного продукту.....	10
1.3.2	Діаграма прецедентів програмного продукту.....	11
1.3.3	Опис сценаріїв використання прецедентів програмного продукту.....	12
1.4	Функціональні вимоги до програмного продукту.....	14
1.4.1.	Багаторівнева класифікація функціональних вимог.....	14
1.4.2	Функціональний аналіз існуючих програмних продуктів.....	15
1.5	Нефункціональні вимоги до програмного продукту.....	16
1.5.1	Опис зовнішніх інтерфейсів.....	16
1.5.1.1	Опис інтерфейса користувача.....	16

					ІС КР 122 АІ-183 ПЗ			
З м	А р	№.	П і д п	Д а				
					ОНПУ, каф. ІС, гр. АІ183			
					Л і т.	Лист	Л и с т і в	
						з	62	

1.5.1.1.1	Опис INPUT-інтерфейса користувача.....	16
1.5.1.1.2	Опис OUTPUT-інтерфейса користувача.....	17
1.5.1.2	Опис інтерфейсу із зовнішніми пристроями.....	18
1.5.1.3	Опис програмних інтерфейсів.....	19
1.5.1.4	Опис інтерфейсів передачі інформації.....	20
1.5.1.5	Опис атрибутів продуктивності.....	20
2	Планування процесу розробки програмного продукту.....	21
2.1	Планування ітерацій розробки програмного продукту.....	21
2.2	Концептуальний опис архітектури програмного продукту.....	22
2.3	План розробки програмного продукту.....	23
2.3.1	Оцінка трудомісткості розробки програмного продукту.....	23
2.3.2	Визначення дерева робіт з розробки програмного продукту...	25
2.3.3	Графік робіт з розробки програмного продукту.....	26
2.3.3.1	Таблиця з графіком робіт.....	26
2.3.3.2	Діаграма Ганта.....	27
3	Проектування програмного продукту.....	28
3.1	Концептуальне та логічне проектування структур даних програмного продукту.....	28
3.1.1	Концептуальне проектування на основі UML-діаграми концептуальних класів.....	28
3.1.2	Логічне проектування структур даних.....	28
3.2	Проектування програмних класів.....	29
3.3	Проектування алгоритмів роботи методів програмних класів...	30
3.4	Проектування тестових наборів методів програмних класів....	32
4	Конструювання програмного продукту.....	40
4.1	Особливості конструювання структур даних.....	40
4.1.1	Особливості інсталяції та роботи з СУБД.....	40

					ІС КР 122 АІ-183 ПЗ			
З м	А р	№.	П і д п	Д а		Л і т.	Лист	Л и с т і в
							5	62
					ОНПУ, каф. ІС, гр. АІ183			

4.2 Особливості конструювання програмних модулів.....	41
4.2.1 Особливості роботи з інтегрованим середовищем розробки.....	41
4.2.2 Особливості створення програмної структури з урахуванням спеціалізованого Фреймворку.....	42
4.2.3 Особливості створення програмних класів.....	43
4.2.4 Особливості розробки алгоритмів методів програмних класів або процедур/функцій.....	45
4.3 Модульне тестування програмних класів.....	46
5 Розгортання та валідація програмного продукту.....	49
5.1 Інструкція з встановлення програмного продукту.....	49
5.2 Інструкція з використання програмного продукту.....	50
5.3 Результати валідації програмного продукту.....	51
Висновки до курсової роботи.....	54
Перелік посилань.....	55

					ІС КР 122 АІ-183 ПЗ			
З м	А р	№.	П і д п	Д а				
					ОНПУ, каф. ІС, гр. АІ183			
					Л і т .	Л и с т	Л и с т і в	
						5	62	

# 1 Вимоги до програмного продукту

## 1.1Визначення потреб споживача

### 1.1.1 Ієрархія потреб споживача

Відомо, що в теорії маркетингу потреби людини можуть бути представлені у вигляді ієрархії потреб ідей американського психолога Абрахама Маслоу включають рівні:

- фізіологія (вода, їжа, житло, сон);
- безпека (особиста, здоров'я, стабільність),
- приналежність (спілкування, дружба, любов),
- визнання (повага оточуючих, самооцінка),
- самовираження (вдосконалення, персональний розвиток).

На рисунку 1.1 представлено одну ієрархію потреби споживача, яку хотілося б задовольнити, використовуючи майбутній програмний продукт.

Майбутній програмний продукт буде задовольняти рівень самовираження, так як сайт створений для навчання і можливістю поділитися своїми знаннями з іншими.



Рис. 1.1 – Приклад ієрархії потреби споживача

## 1.2 Деталізація матеріальної потреби

Для деталізації матеріальної потреби були використані ментальні

карти (MindMap). При створенні ментальних карт матеріальна потреба розташовується в центрі карти. Асоціативні гілки були швидко створені, припускаючи, що в загальному вигляді з об'єктом пов'язані три потоки даних інформації: вхідний, внутрішній, вихідний. Кожен потік - це асоціативна група, що включає можливі гілки, що відповідають на п'ять питань: Хто? Що? Де? Коли? Як? Відповідно до рекомендацій по створенню ментальних карт кожна гілка-асоціація розділена на додаткові асоціативні гілки, які деталізують відповіді на поставлені питання.

На рисунку 1.2 представлено приклад деталізації матеріальної потреби.

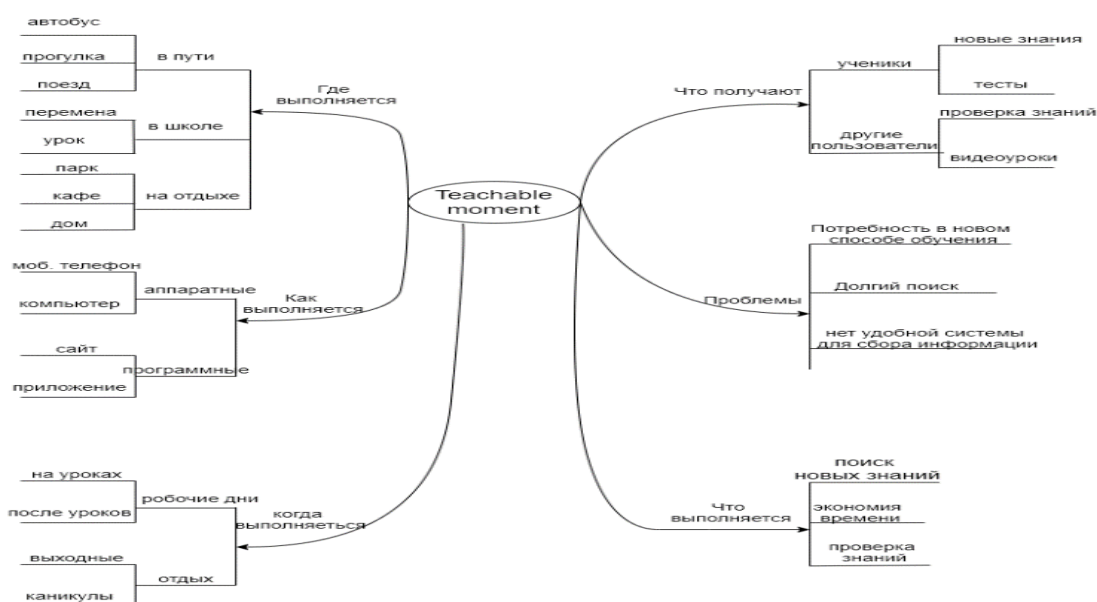


Рис. 1.2 – Приклад деталізації матеріальної потреби

## 1.2 Бізнес-вимоги до програмного продукту

### 1.2.1 Опис проблеми споживача

#### 1.2.1.1 Концептуальний опис проблеми споживача

Для скорочення часу і коштів при задоволенні реальних потреб людині потрібна інформація, що призводить до появи інформаційної потреби. В таблиці 1 представлено опис проблеми споживача та метричні показники незадоволеності споживача.

					ІС КР 122 АІ-183 ПЗ	Л и
Змін	Л и	№.	П і д п	Д а		7



<b>Загальний опис проблеми</b>	<b>Метричні показники незадоволеності споживача</b>
Немає зручних мереж в яких одна людина, яка не розуміє матеріал, може швидко та безкоштовно отримати допомогу від іншої, яка в цій темі має досвід	Низький відсоток людей, які хочуть вивчити новий матеріал, змогли знайти, де це зробити.

Таблиця 1 – опис проблеми споживача та метричні показники незадоволеності споживача

#### **1.2.1.2 Метричний опис проблеми споживача**

Рівень доступності AL (AL – Access Level) можна визначити як  $AL = NA / N$ , де NA – кількість пройденого матеріалу; N – загальна кількість людей які хочуть вивчити нову інформацію.

#### **1.2.2 Мета створення програмного продукту**

##### **1.2.2.1 Проблемний аналіз існуючих програмних продуктів**

Проблемний аналіз існуючих програмних продуктів включає кроки:

- 1) формування переліку товарів через пошук в інтернеті;
- 2) формування таблиці рішення проблем (рядки - назви продуктів, стовпці - проблеми, осередки - позначки про рішення проблем продуктом);

В таблиці 2 представлено таблицю програмного аналізу існуючих програмних продуктів.

N	Назва продукту	Вартість		Ступінь готовності	Примітка
1	<b>Zhihu.com</b>	Безкоштовно		1	1) Не локалізований. 2) Для реєстрації потрібен номер телефона. 3) Немає можливості самому створювати новий курс 4) Немає можливості додавати свої відео-курси та тести
2	<b>Quora.com</b>	Безкоштовно		1	1) Немає можливості обрати російську або українську мову сайту. 2) Немає можливості самому створювати новий курс 3) Немає можливості додавати свої відео-курси та тести

Таблиця 2 – реалізація програмного аналізу існуючих програмних продуктів

					ІС КР 122 АІ-183 ПЗ	Л и
Змін	Л и	№.	П і д п	Д а		9

### 1.2.2.2 Мета створення програмного продукту

Метою створення програмного продукту є підвищення відсотка людей, які змогли знайти курс який їм підходить, щоб вивчити новий матеріал.

### 1.2.3 Назва програмного продукту

#### 1.2.3.1 Гасло програмного продукту

Відмінним способом представлення назви є його логотип, що поєднують зорові образи і короткі фрази-гасла.

Гаслом даного програмного продукту є - «Сайт-сервіс для навчання – Teachable moment».

#### 1.2.3.2 Логотип програмного продукту

Логотип – це графічний знак, емблема або символ, який використовується територіальними утвореннями, комерційними підприємствами, організаціями та приватними особами для підвищення впізнаваності і розпізнаваності в соціумі. Логотип являє собою назву сутності, яку він ідентифікує, у вигляді стилізованих букв або ідеограми.

На рисунку 1.2.3 представлено логотип ПП:



Рис. 1.2.3. – логотип ПП

### 1.3 Вимоги користувача до програмного продукту

#### 1.3.1 Історія користувача програмного продукту

Створення User Story споживача ПП

					ІС КР 122 АІ-183 ПЗ	Л и
Змін	Л и	№.	П і д п	Д а		10

- 1. Як гість я можу зареєструватись в системі для отримання облікового запису для користування додатком.
- 2. Як гість я можу зайти в створений раніше акаунт.
- 3. Як користувач я можу проходити відео-уроки.
- 4. Як користувач я можу проходити тести після отриманої інформації.
- 5. Як користувач я можу викладати нові відео-уроки та тести для проходження їх іншими.

### 1.3.2 Діаграма прецедентів програмного продукту

Діаграма прецедентів (Use Case UML-діаграма) включає:

- актори (зацікавлені особи і зовнішні системи зі своїм API);
- прецеденти як основні функції ПП;
- зв'язки між прецедентами і акторами як множиною зацікавлених осіб;

На рисунку 1.3 представлено діаграму прецедентів ПП.

					ІС КР 122 АІ-183 ПЗ	Л и
Змін	Л и	№.	П і д п	Д а		11



Рис 1.3 – діаграма прецедентів ПП

### 1.3.3 Опис сценаріїв використання прецедентів програмного продукту

Опис сценаріїв:

1. А

вАктори, що зацікавлені в виконанні даного прецеденту:

џКористувач;

еГість.

Актор-основа початку: Гість.

иГарантії успіху: успішна авторизація в додаток.

з Успішний сценарій:

а. ПП запитує у користувача параметри авторизації;

ц

і

					ІС КР 122 АІ-183 ПЗ	Л и
		я				12
Змін	Л и	№.	П і д п	Д а		

2. Користувач передає свої параметри;
3. ПП надає доступ.

Альтернативний сценарій:

1. Користувач повідомляє неправильні дані;
2. ПП виводить повідомлення про помилку

## 2. Здобути нові знання:

- Умова початку прецеденту: прецедент «Здобути нові знання».
- Актори, що зацікавлені в виконанні даного прецеденту: Користувач.
- Актор-основа початку: Користувач.
- Гарантії успіху: успішне здобуття нових знань користувача ПП.

Успішний сценарій:

1. Користувач знаходить корисну інформацію.
2. ПП показує відеоуроки для нових знань;

Альтернативний сценарій:

1. Користувач не може почати навчання;
2. ПП виводить повідомлення про помилку.

## 3. Перевірка тестами пройденого матеріалу:

- Умова початку прецеденту: прецедент «Здобути нові знання».
- Актори, що зацікавлені в виконанні даного прецеденту: Користувач.
- Актор-основа початку: Користувач.
- Гарантії успіху: успішно пройдений тест після здобуття нових знань користувача ПП.

Успішний сценарій:

1. Користувач проходить тести для перевірки пройденого матеріалу.
2. Користувач має можливість засвоїти матеріал;

Альтернативний сценарій:

1. Користувач не має можливості тестування;

					ІС КР 122 АІ-183 ПЗ	Л и
						13
Змін	Л и	№.	П і д п	Д а		

2. ПП виводить повідомлення про помилку.

4. Створити свій курс:

- Умова початку прецеденту: прецедент «Поділитися знаннями».
- Актори, що зацікавлені в виконанні даного прецеденту: Користувач.
- Актор-основа початку: Користувач.
- Гарантії успіху: успішно створений курс.

Успішний сценарій:

1. Користувач може створити новий курс.

2. Користувач додає новий курс;

Альтернативний сценарій:

1. ПП отримує неправильні данні від користувача;

2. ПП виводить повідомлення про помилку.

## 1.4 Функціональні вимоги до програмного продукту

### 1.4.1. Багаторівнева класифікація функціональних вимог

З метою упорядкування функцій ПП створено багаторівневу класифікацію функціональних вимог (Functional Requirements - FR), виявлених в сценаріях використання прецедентів.

В таблиці 3 представлено таблицю класифікації функціональних вимог.

Ідентифікатор функції	Назва функції
FR 1	Авторизація
FR 1.1	Запитання у користувача параметрів авторизації та передача користувачем своїх параметрів з подальшим наданням доступу
FR 1.2	Отримання неправильних параметрів від користувача та виведення повідомлення про помилку

FR 2	Перевірка тестами пройденого матеріалу
FR 2.1	Показ тестів для перевірки пройденого матеріалу та допомога засвоєнню матеріалу
FR 2.2	Неможливість з боку ПП надати можливість тестування з подальшим виведенням повідомлення про помилку
FR 3	Створення свого курсу
FR 3.1	Надання форми на заповнення для створення нового курсу з подальшим додаванням створеного курсу
FR 3.2	Отримання неправильних даних від користувача з подальшим виведенням повідомлення про помилку

Таблиця 3 - результат класифікації функціональних вимог

#### 1.4.2 Функціональний аналіз існуючих програмних продуктів

В таблиці 4 представлено функціональний аналіз існуючих програмних продуктів.

Ідентифікатор функції	Zhihu.com	Quora.com
FR 1.1	+	+
FR 1.2	+	+
FR 2.1	-	+
FR 2.2	-	-
FR 3.1	-	-
FR 3.2	-	-

Таблиця 4 - функціональний аналіз існуючих програмних продуктів



## 1.5 Нефункціональні вимоги до програмного продукту

### 1.5.1 Опис зовнішніх інтерфейсів

#### 1.5.1.1 Опис інтерфейса користувача

##### 1.5.1.1.1 Опис INPUT-інтерфейса користувача

Результат аналізу засобів INPUT-потоків представлено у вигляді таблиці 5

Ідентифікатор функції	Засіб INPUT-потoku	Особливості використання
FR 1.1	Стандартна комп'ютерна клавіатура; 2/3-кнопочний маніпулятор типу "миша"; сенсорний екран (Touchscreen, Touchpad, Multi-touch);	Ліва кнопка миші – відправка запиту на авторизацію на сайт
FR 1.2		-
FR 2.1		Ліва кнопка миші – відправка зроблених тестів на сайт для перевірки
FR 2.2		-
FR 3.1		Ліва кнопка миші – відправка на сайт запиту для створення нового курсу
FR 3.2		-

Таблиця 5 - опис INPUT- інтерфейса користувача

##### 1.5.1.1.2 Опис OUTPUT-інтерфейса користувача

Результат аналізу засобів OUTPUT-потоків представлено у вигляді таблиці 6.

Ідентифікатор функції	Засіб OUTPUT-поток
FR 1.1	<p><b>Авторизация</b></p> <p>Имя пользователя Placeholder</p> <p>Пароль Placeholder</p> <p><a href="#">Забыли пароль?</a></p> <p><input checked="" type="checkbox"/> Запомнить меня <input type="button" value="Войти"/></p>
FR 1.2	<p><b>Авторизация</b></p> <p>Имя пользователя Placeholder</p> <p>Пароль Placeholder</p> <p>Неверное имя или пароль!!!!!!!!!!!!</p> <p><a href="#">Забыли пароль?</a></p> <p><input checked="" type="checkbox"/> Запомнить меня <input type="button" value="Войти"/></p>
FR 2.1	<p><b>Тест</b></p> <p>Що таке відстань до об'єкта</p> <p> <input type="radio"/> Бісектриса <input type="radio"/> Медіана <input type="radio"/> Парабола <input type="radio"/> Перпендикуляр </p> <p>Дискримінант дорівнює</p> <p> <input type="radio"/> <math>2 \cdot a \cdot b</math> <input type="radio"/> <math>b^2 - 4ac</math> <input type="radio"/> <math>x^2 - y^2</math> <input type="radio"/> <math>b^2 - ac</math> </p> <p><math>x^2 - 5x + 6 = 0</math></p> <p> <input type="radio"/> <math>x_1=2; x_2=3</math> <input type="radio"/> <math>x=0</math> <input type="radio"/> Відповіді немає <input type="radio"/> <math>b^2 - ac</math> </p> <p><input type="button" value="Опубликовать"/></p>

FR 2.2	<p><b>Тест</b></p> <p>Що таке відстань до об'єкта</p> <p> <input type="radio"/> Бісектриса <input type="radio"/> Медіана <input type="radio"/> Парабола <input type="radio"/> Перпендикуляр </p> <p>Дискримінант дорівнює</p> <p> <input type="radio"/> <math>2^2 a^2 b</math> <input type="radio"/> <math>b^2 - 4ac</math> <input type="radio"/> <math>x^2 - y^2</math> <input type="radio"/> <math>b^2 - ac</math> </p> <p><math>x^2 - 5x + 6 = 0</math></p> <p> <input type="radio"/> <math>x_1=2; x_2=3</math> <input type="radio"/> <math>x=0</math> <input type="radio"/> Відповіді немає <input type="radio"/> <math>b^2 - ac</math> </p> <p>Опублікувати</p>
FR 3.1	<p><b>Создание нового курса</b></p> <p>Название курса: Категория:</p> <p>Ссылка на видеоролик:</p> <p>Опублікувати</p>
FR 3.2	<p><b>Создание нового курса</b></p> <p>Название курса: Категория:</p> <p>Ссылка на видеоролик:</p> <p>Опублікувати</p>

Таблиця 6 - результат аналізу засобів OUTPUT-потоків

### 1.5.1.2 Опис інтерфейсу із зовнішніми пристроями

Для опису інтерфейсів із зовнішніми пристроями використано такі обладнання:

- Desktop-персональний комп'ютер;

- Notebook;
- смартфон;
- мобільний телефон;
- планшет.

В таблиці 7 представлено таблицю опису інтерфейсу із зовнішніми пристроями.

Ідентифікатор функції	Зовнішній пристрій
FR 1.1 – FR 1.2	Смартфон, планшет, Desktop-персональний комп'ютер, Notebook;
FR 2.1 – FR 2.2	
FR 3.1 – FR 3.2	

Таблиця 7 – опис інтерфейсу із зовнішніми пристроями

### 1.5.1.3 Опис програмних інтерфейсів

Версії операційних систем, які знадобляться при реалізації більшості функцій ПП:

- Linux
- Windows
- Android
- IOS
- Підтримка браузерів з html 5 та JS

Бібліотеки:

- Axios
- Router
- VueSweetalert2
- Mongoose
- body-parse
- CORS

#### 1.5.1.4 Опис інтерфейсів передачі інформації

Опис інтерфейсів передачі інформації, які знадобляться при реалізації більшості функцій ПП:

Провідні інтерфейси:

- Ethernet

Безпроводні інтерфейси:

- Wi-Fi;

#### 1.5.1.5 Опис атрибутів продуктивності

Із множини атрибутів якості в роботі розглянуто лише продуктивність.

Найчастіше використовувані такі характеристики продуктивності:

- максимальний час реакції ПП на дії користувачів;
- максимальна кількість одночасно обслуговуваних користувачів.

В роботі розглянуто лише максимальний час реакції ПП на дії користувачів.

В таблиці 8 представлено результат аналізу характеристик продуктивності.

Ідентифікатор функції	Максимальний час реакції ПП на дії користувачів, секунди
FR 1.1	3
FR 1.2	3
FR 2.1	2
FR 2.2	2
FR 3.1	3
FR 3.2	3

Таблиця 8 – результат аналізу характеристик продуктивності

## 2 Планування процесу розробки програмного продукту

### 2.1 Планування ітерацій розробки програмного продукту

При створенні пріоритетів враховано:

- сценарні залежності між прецедентами, до яких належать функції, на основі аналізу пунктів передумов початку роботи прецедентів, вказаних в описі сценаріїв роботи прецедентів;
- вплив роботи прецеденту, до якого належить функція, на досягнення мети ПП, наприклад у відсотках, на основі аналізу пунктів гарантій успіху, вказаних в описі сценаріїв роботи прецедентів.

Сценарні залежності будуть перетворені у відповідні функціональні залежності.

Вплив роботи прецеденту поширено на всі підлеглі функції ієрархії.

При визначенні пріоритетів використано наступні позначки:

- М (Must) – функція повинна бути реалізованою у перших ітераціях за будь-яких обставин;
- S (Should) – функція повинна бути реалізованою у перших ітераціях, якщо це взагалі можливо;
- С (Could) – функція може бути реалізованою, якщо це не вплине негативно на строки розробки;
- W (Want) – функція може бути реалізованою у наступних ітераціях.

Приклад опису представлено в таблиці 9.

					ІС КР 122 АІ-183 ПЗ	Л и
						21
Змін	Л и	№.	П і д п	Д а		

Ідентифікатор функції	Назва функції	Функціональні залежності	Вплив на досягнення мети	Реалізація функції
FR 2.1	Перевірка тестами пройденого матеріалу	-	30%	M
FR 2.2		FR 2.1	0%	M
FR 3.1	Створення свого курсу	-	65%	M
FR 3.2		FR 3.1	0%	M
FR 1.1	Авторизація	-	5%	S
FR 1.2		FR 1.1	0%	S

Таблиця 9 - планування ітерацій розробки програмного продукту

## 2.2 Концептуальний опис архітектури програмного продукту

Компоненти ПП, відповідальні за три рівня роботи, які історично називають:

– рівень уявлення – Presentation Level (PL), який містить компоненти зовнішнього інтерфейсу (програмний інтерфейс користувача), наприклад, Client Operation System, Input, Output;

– рівень бізнес-логіки – Business Level (BL), який містить апаратно-програмні компоненти обробки даних, наприклад, Server Operation System, Application Server;

– рівень зберігання даних – Access Level (AL), який містить апаратно-програмні компоненти керування даними, наприклад, Server Operation System, DataBase Server.

В сучасній термінології програмні компоненти PL-рівня називають FrondEnd, а

програмні компоненти BL-рівня та AL-рівня називають Backend.

Для кожного компонента вказано особливості реалізації, наприклад: типи апаратних компонентів, операційні системи, програмні технології розробки, системи керування базами даних, типи структур даних (реляційні, XML, JSON).

На рисунку 2.2 представлено концептуальний опис архітектури програмного продукту.

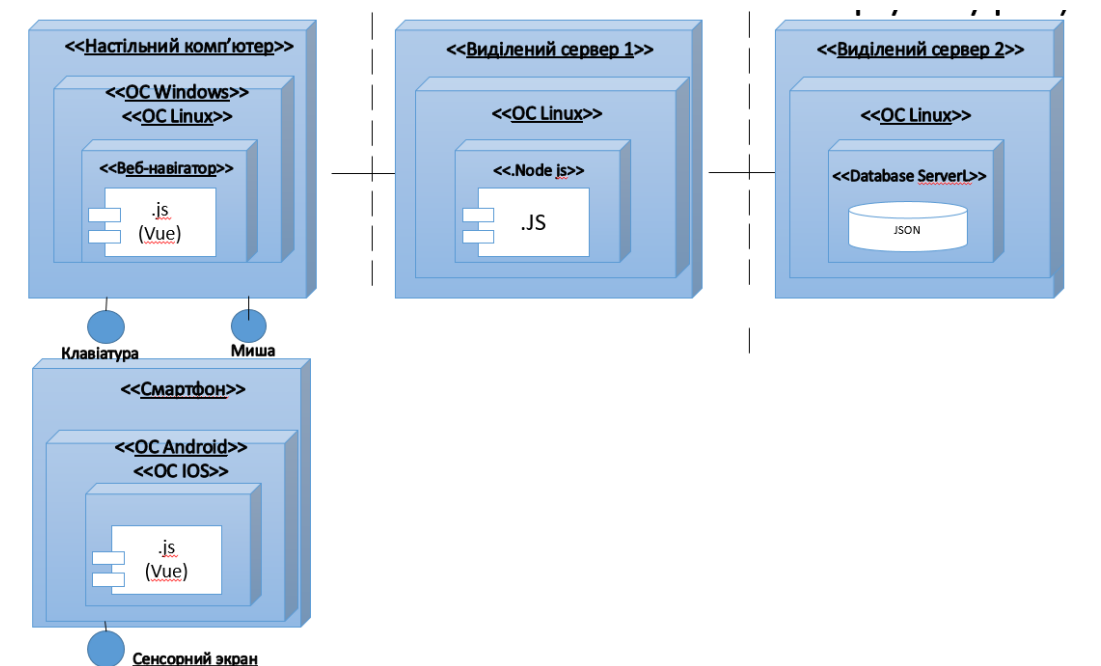


Рис 2.2 – концептуальний опис архітектури програмного продукту.

## 2.3 План розробки програмного продукту

### 2.3.1 Оцінка трудомісткості розробки програмного продукту

#### 2.3.1.1 Всі актори діляться на три типи: прості, середні і складні.

Простий актор представляє зовнішню систему з чітко визначеним програмним інтерфейсом. Середній актор представляє або зовнішню систему, що взаємодіє з ПП за допомогою мережевих протоколів, або особистість, що користується текстовим інтерфейсом (наприклад, алфавітно-цифровим терміналом). Складний актор представляє особистість, що користується графічним інтерфейсом.



Загальна кількість акторів кожного типу помножується на відповідний ваговий коефіцієнт, потім обчислюється загальний ваговий показник (таблиця 10 ).

Назва актора	Тип актора	Ваговий коефіцієнт
Користувач	Складний	3
Гість	Легкий	1

Таблиця 10 – вагові коефіцієнти акторів

### 2.3.1.2 Визначення вагових показників прецедентів UC

Всі прецеденти діляться на три типи; прості, середні і складні в залежності від кількості кроків успішних сценаріїв (основних і альтернативних).

Загальна кількість прецедентів кожного типу помножується на відповідний ваговий коефіцієнт, потім обчислюється загальний ваговий показник (таблиця 11).

Назва прецедента	Тип прецедента	Кількість кроків сценарію	Ваговий коефіцієнт
FR1	Простий	$\leq 3$	5
FR2	Середній	4-7	7
FR2	Складний	$> 7$	15

Таблиця 11 – вагові коефіцієнти прецедентів

UUCP=20

### 2.3.1.3 Визначення технічної складності проекту

Технічна складність проекту (TCF - Technical Complexity Factor) обчислюється з урахуванням показників технічної складності (таблиця 12).

Кожному показнику присвоюється значення STi в діапазоні від 0 до 5:

- 0 означає відсутність значимості показника для даного проекту,

- 5 - високу значимість.

					IC KP 122 AI-183 ПЗ	Л и
						24
Змін	Л и	№.	П і д п	Д а		

Значения TCF обчислюється за формулою:

$$TCF = 0,6 + (0,01 * (ST_i * Вага_i))$$

Показник	Опис показника	ST <sub>i</sub>	Вага	STF
T1	Распределенная система	1	0.5	0.605
T2	Высокая производительность (пропускная способность)	3	1	0.63
T3	Работа конечных пользователей в режиме онлайн	5	1	0.65
T4	Сложная обработка данных	0.5	0.5	0.6025
T5	Повторное использование кода	0.5	0.5	0.6025
T6	Простота установки	1	0.5	0.605
T7	Простота использования	3	2	0.66
T8	Переносимость	1	0.5	0.605
T9	Простота внесения изменений	1	2	0.62
T10	Параллелизм	0	0	0.6
T11	Специальные требования к безопасности	1	0.5	0.605
T12	Непосредственный доступ к системе со	1	0.5	0.605

	сторони внешних пользователей			
T13	Специальные требования к обучению пользователей	1	0.5	0.605

Таблиця 12 - показники технічної складності проекту TCF

$$TCF = 7.995$$

$$EF = 10.36$$

$$UCP = UUCP * TCF * EF = 20 * 7.995 * 10.36 = 1656.564$$

### 2.3.2 Визначення дерева робіт з розробки програмного продукту

Кожна функція 1-го рівня ієрархії перетворюється в Work Package (WP)

Кожна функція 2-го рівня ієрархії перетворюється в Work Task (WT).

Для кожної задачі визначаються підзадачі - Work SubTask (WST) з

урахуванням базових процесів розробки програмних модулів: проектування, конструювання, модульне тестування, збірка та системне тестування.

Приклад WBS представлено на рисунку 2.3.2

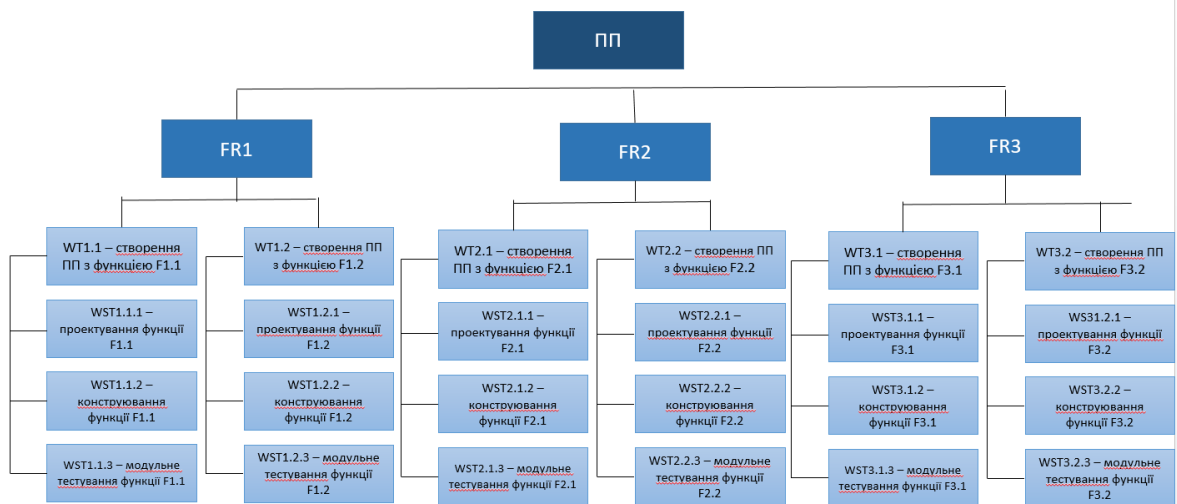


Рис. 2.3.2 – Приклад WBS

### 2.3.3 Графік робіт з розробки програмного продукту

#### 2.3.3.1 Таблиця з графіком робіт

Для кожної підзадачі визначається виконавець, що фіксується у вигляді таблиці, приклад якої представлено в таблиці 13.

WSP	Дата початку	Дні	Дата завершення	Виконавець
1.1	07.10.2020	2	08.10.2020	Ткаченко К.І.
1.2	08.10.2020	2	09.10.2020	Кеосак А.І.
2.1	09.10.2020	3	11.10.2020	Залуковский Ю.О.
2.2	11.10.2020	1	11.10.2020	Гірля А.І.
3.1	12.10.2020	3	14.10.2020	Ткаченко К.І.
3.2	14.10.2020	2	14.10.2020	Кеосак А.І.

Таблиця 13 - приклад опису підзадач із закріпленням виконавців

#### 2.3.3.2 Діаграма Ганта

Діаграма Ганта (складається із смуг (вісь Y), орієнтованих уздовж осі часу (вісь X)). Кожна смуга – окрема підзадача в проекті, її кінці - моменти початку і завершення роботи, її протяжність - тривалість роботи. Мета діаграми - візуально показати послідовність процесів та можливість паралельного виконання робіт. Для створення діаграми Ганта створено таблицю з графіком робіт, приклад якої представлено на рисунку 2.3.3

	07.окт	08.окт	09.окт	10.окт	11.окт	12.окт	13.окт	14.окт
WSP								
1 1	Ткаченко							
1 2		Кеосак						
2 1			Залуковский					
3 1					Гірля			
3 1						Ткаченко		
3 2								Кеосак

Рис. 2.3.3 – Приклад таблиці графіка робіт

### 3 Проектування програмного продукту

#### 3.1 Концептуальне та логічне проектування структур даних програмного продукту

##### 3.1.1 Концептуальне проектування на основі UML-діаграми концептуальних класів

Використовуючи кроки основного успішного та альтернативного сценаріїв роботи прецедентів ПП, спроектовано UML-діаграму концептуальних класів.

Моделювання проведено з урахуванням наступної послідовності кроків.

1. Визначено імена класів: Користувач, Курс, Тест, Запитання, Відповідь.
  2. Визначено імена атрибутів класів: Логін, Пароль, Пошта, Відео, Опис, Назва, Завдання, Правильна відповідь.
  3. Визначено зв'язку між класами: 1 до N, N до N.
  4. Створено UML-діаграму класів, яку представлено на рисунку 3.1
- представлено UML-діаграму класів.

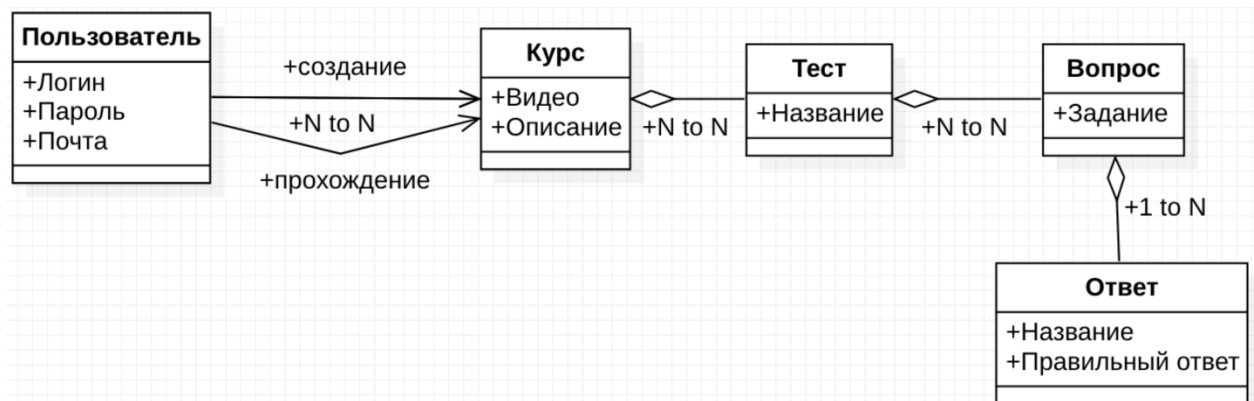


Рис. 3.1 - UML-діаграма класів

### 3.1.2 Логічне проектування структур даних

UML-діаграму концептуальних класів перетворено в опис структур даних з використанням моделі, яка була обрана в концептуальному описі архітектури ПП. На рисунку 3.1.2 представлено дану діаграму.

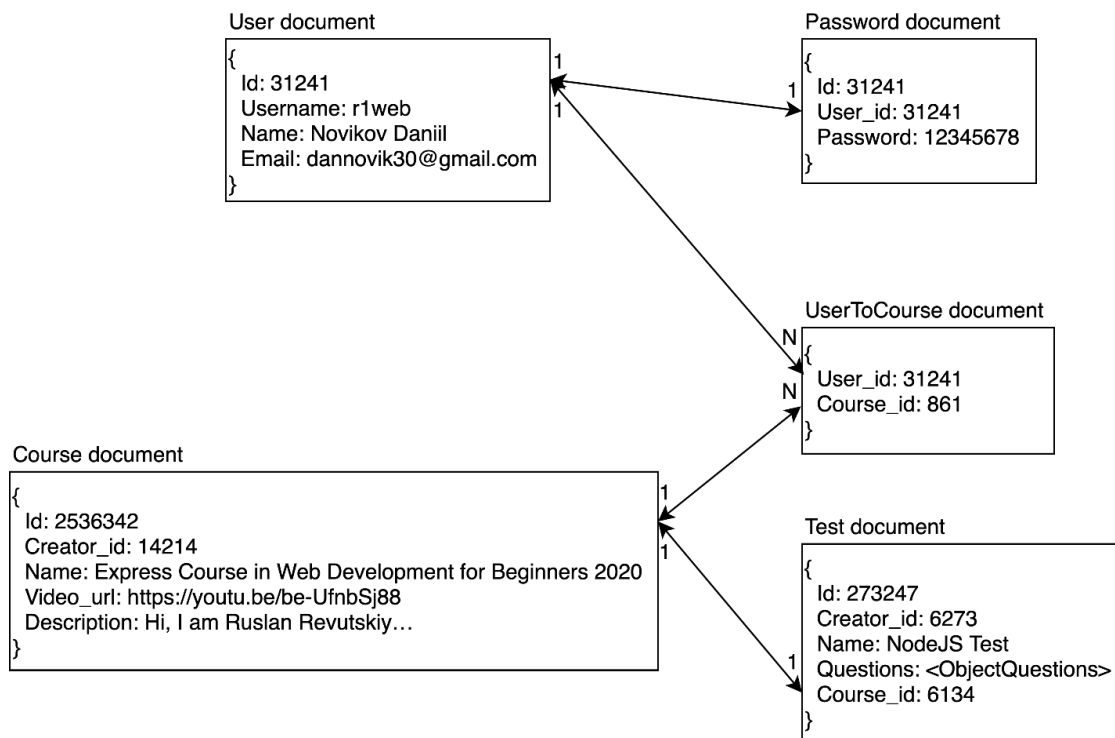


Рис. 3.1.2 – діаграма логічного проектування даних

### 3.2 Проектування програмних класів

На основі UML-діаграми концептуальних класів спроектовано програмні класи, визначивши:

- англійські або транслітерацію україномовних назв класів та їх атрибутів;
- абстрактні класи, їх класи-нащадки та інші класи;
- зв'язки між класами та їх кратності;
- атрибути класів с типами даних (цілий, дійсний, логічний, перелічуваний, символьний).
- методи-конструктори ініціалізації екземплярів об'єктів класу, set-методи та get-методи для доступу до атрибутів класу.

На рисунку 3.2 представлено проектування UML-діаграми програмних класів.

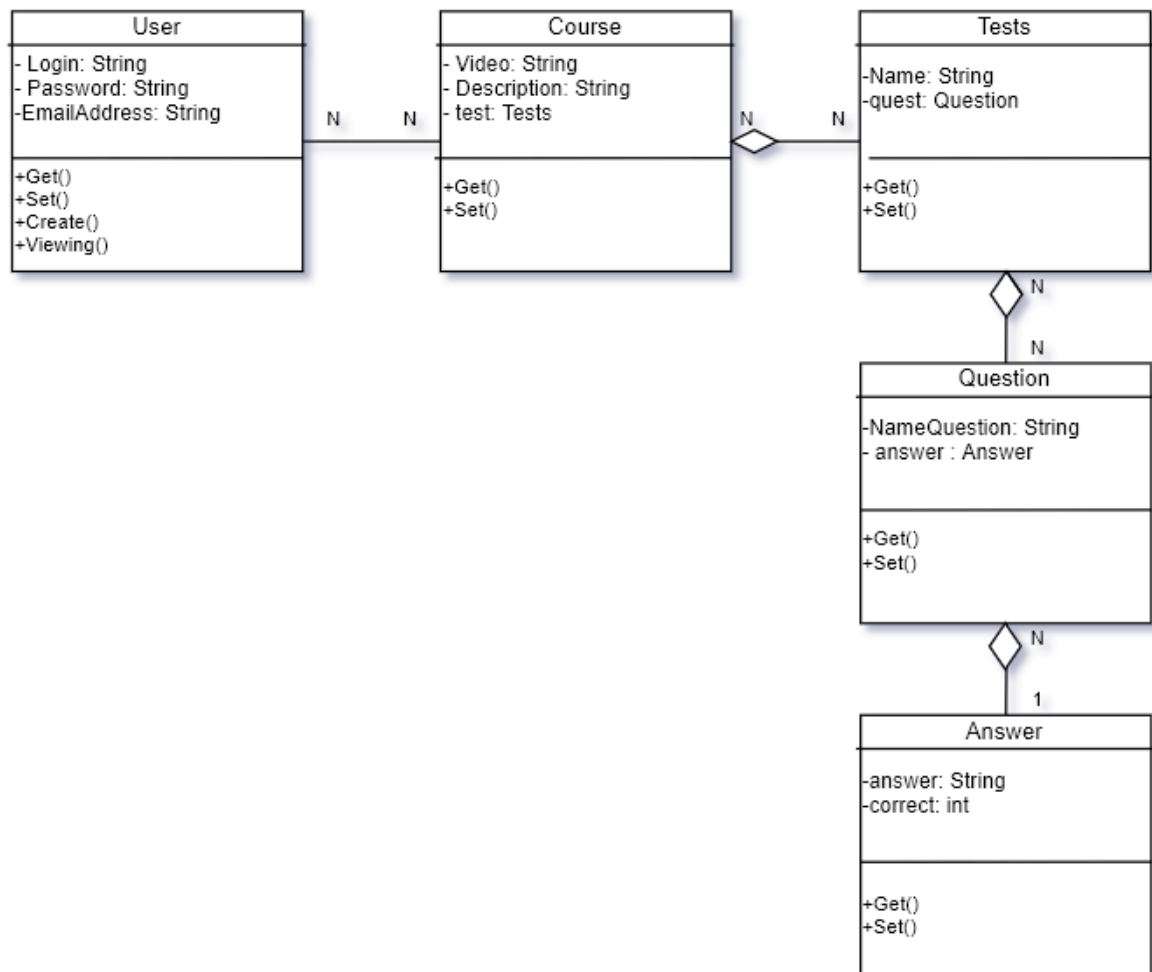


Рис 3.2 - проектування UML-діаграми програмних класів

### 3.3 Проектування алгоритмів роботи методів програмних класів

З усіх методів виділені ті, що мають операції доступу до БД або містять керуючі умови (if-then-else, while).

Описано алгоритм роботи у вигляді UML-діаграми активності:

– кожен опис алгоритму представлений на мові PlantUML, використовуючи веб-редактор;

Далі представлений результат роботи проектування алгоритмів роботи методів програмних класів.

Метод saveUser():

Код програми:

```
@startuml
title saveUser()
(*) --> "Enter parameters of User"
    --> " Sending data to the database as a json file with fields
id, login, password, email "
    --> if "User is saved"
        --> [Success] "Choose one User of saved Users"
        --> "Display success choosing"
    --> "Choose output Users"
        --> (*)
    else
        --> [Failure] "Display error message"
    endif
    --> (*)
@enduml
```

					ІС КР 122 АІ-183 ПЗ	Л и
Змін	Л и	№.	П і д п	Д а		31



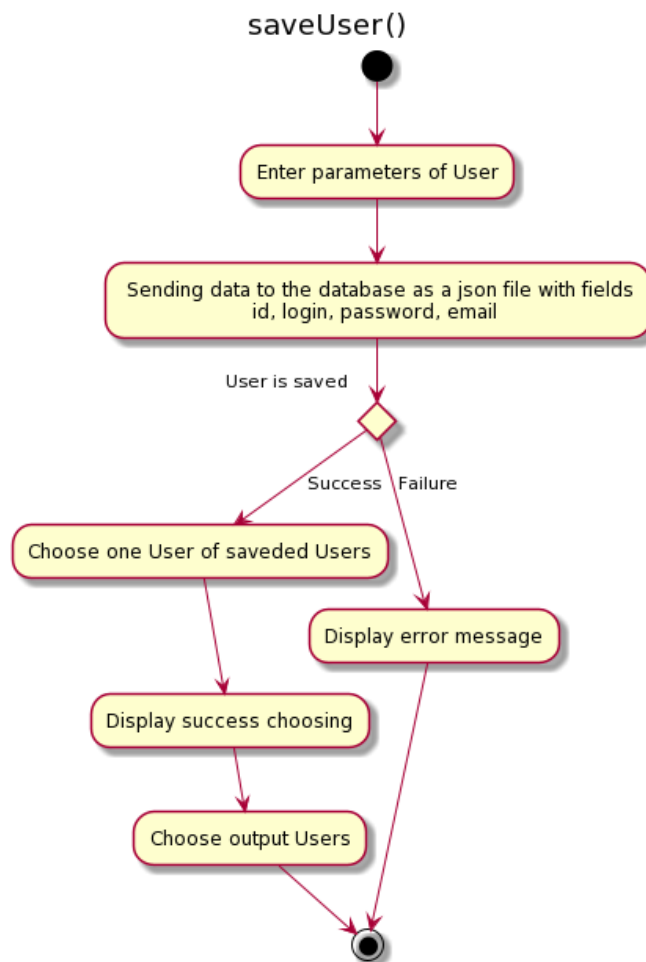


Рис. 3.3.1 UML - діаграма активності методу saveUser.

### 3.4 Проектування тестових наборів методів програмних класів

Принцип тестування чорного ящика розглядає програму або окремий програмний модуль як чорний ящик без вивчення її вмісту у вигляді алгоритму її робіт, інакше розглядається принцип тестування білого ящика.

Метою тестування методами чорного ящика є перевірка умов, при яких поведінка програми або програмного модуля не відповідає її специфікації. Для виявлення всіх помилок в програмі необхідно виконати вичерпне тестування, тобто тестування на всіляких наборах вхідних даних. Для більшості програм таке неможливо, тому застосовують розумне тестування, при якому тестування програми обмежується невеликою підмножиною всіляких наборів даних. При цьому необхідно вибрати найбільш підходящі підмножини, підмножини з найвищою імовірністю виявлення помилок.

Результат представлений в таблиці 14.

Назва функції	№ тесту	Опис значень вхідних даних	Опис очікуваних значень результату
addUser()	1	[]	{ “status”: 404, “message”: “Error! Parameters cannot be empty” }
	2	{ “Login”: “12234re2r” }	{ “status”: 404, “message”: “Error! Parameters cannot be empty” }
	3	{ “Login”: “12234re2r”, “Password”: “123423re2r” }	{ “status”: 404, “message”: “Error! Parameters cannot be empty” }
	4	{ “Email”: “12.com” }	{ “status”: 404, “message”: “Error! Mail entered incorrectly” }
	5	{ “Login”: “12” }	{ “status”: 404, “message”: “Errors! Parameters is less than 4 characters” }
	6	{ “Login”: “12234re2r”, “Password”: “123423re2r”, “Email”: “12@gmail.com” }	{ “status”: 202, “message”: “Successful! User add” }
saveUser()	7	[]	{ “status”: 404, “message”: “Error! DB received no data” }
	8	{ “Login”: “12234re2r”, “Password”: “123423re2r”, “Email”: “12@gmail.com” }	{ “status”: 202, “message”: “Successful! User save” }
getData()	1	[]	{

			{ "status": 404, "message": "Error! Parameters cannot be empty" }
	2	{ "Login": "12234re2r" }	{ "status": 404, "message": "Error! Parameters cannot be empty" }
	3	{ "Login": "12234re2r", "Password": "123423re2r" }	{ "status": 404, "message": "Error! Parameters cannot be empty" }
	4	{ "Login": "12" }	{ "status": 404, "message": "Errors! Parameters is less than 4 characters" }
	5	{ "Login": "12234re2r", "Password": "123423re2r" }	{ "status": 202, "message": "Successful! User add" }
setLogin()	1	login = "213554"	// login = "" { "status": 404, "message": "Error! the values are incorrect!" }
	2	login = "213554"	// login = "213554" { "status": 202, "message": "Successful!" }
setPassword()	1	password = "213554"	// password = "" { "status": 404, "message": "Error! the values are incorrect!" }
	2	password = "213554"	// password = "213554" { "status": 202, "message": "Successful!" }

setEmail()	1	email = "21@gmail.com"	// password = "" { "status": 404, "message": "Error! the values are incorrect!" }
	2	email = "21@gmail.com"	// email = "21@gmail.com" { "status": 202, "message": "Successful!" }
Validation()	1	{ "Login": "12234re2r", "Password": "123423re2r" }	// "Login": "lad", // "Password": // "12342" { "status": 404, "message": "Error! there is no such user" }
	2	{ "Login": "lad", "Password": "12342" }	// "Login": "lad", // "Password": // "12342" { "status": 202, "message": "Successful! you entered" }
createCourse()	1	{ "video_url": "youtu.be/sdfiojse", "description": "React Course", "test_id": "1231", "title": "Web" }	{ "result": "course created", "course_model": { "video_url": "youtu.be/sdfiojse", "description": "React Course", "test_id": "1231", "title": "Web" } }
	2	{ "video_url": "youtu.be/sdfiojse", "description": "React Course", "test_id": "1231" }	{ "result": "Course not created. Insufficient input." }
	3	{ "video_url": "youtu.be/sdfiojse", "description": "React Course" }	{ "result": "Course not created. Insufficient input." }

	4	{ "video_url": "youtu.be/sdfiojse" }	{ "result": "Course not created. Insufficient input." }
	5	{ }	{ "result": "Course not created. Insufficient input." }
saveCourse()	1	{ "course_model": { "video_url": "youtu.be/sdfiojse", "description": "React Course", "test_id": "1231", "title": "Web" } }	{ "result": "course added to the database", }
	2	{ }	{ "result": "The course has not been added to the database. Insufficient input.", }
getData()	1	{ "course_id": "2313" }	{ "video_url": "youtu.be/sdfiojse", "description": "React Course", "test_id": "1231", "title": "Web" }
	2	{ }	{ "result": "Insufficient input." }
getVideo()	1	{ "course_id": "2313" }	{ "video_url": "youtu.be/sdfiojse" }
	2	{ }	{ "result": "Insufficient input." }
getDescription()	1	{ "course_id": "2313" }	{ "description": "React Course" }
	2	{ }	{ "result": "Insufficient input." }
getTest()	1	{	{

		“course_id”: “2313” }	“test_id”: “1231” }
	2	{ }	{ “result”: “Insufficient input.” }
setVideo()	1	{ “course_id”: “2313”, “video_url”: “youtu.be/sdfiojse” }	{ “result”: “Video is not set. Insufficient input.” }
	2	{ }	{ “result”: “Video is not set. Insufficient input.” }
setDescription()	1	{ “course_id”: “2313”, “description”: “React Course” }	{ “result”: “description is set” }
	2	{ }	{ “result”: “Description is not set. Insufficient input.” }
setTest()	1	{ “course_id”: “2313”, “test_id”: “1231” }	{ “result”: “Test is set” }
	2	{ }	{ “result”: “Test is not set. Insufficient input.” }
Viewing()	1	{ “course_id”: “2313” }	{ “result”: { “video_url”: “youtu.be/sdfiojse”, “description”: “React Course”, “test_id”: “1231”, “title”: “Web” } }
	2	{ }	{ }

			“result”: “Course does not exist” }
getName()	1	{ “test_id”: “1231” }	{ “result”: { “name”: “test for the course "Web" } }
	2	{ }	{ “result”: “Test does not exist” }
getQuestion()	1	{ “question_id”: “1” }	{ “result”: { “question”: “Question1”, “answer_id”: “1” } }
	2	{ }	{ “result”: “The question does not exist” }
setName()	1	{ “name”: “Test1” }	{ “result”: “The name is set” }
	2	{ }	{ “result”: “Name is not set. Insufficient input.” }
getNameQuestion()	1	{ “question_id”: “1” }	{ “result”: “Name question” }
	2	{ }	{ “result”: “Insufficient input.” }
getAnswer()	1	{ “question_id”: “1” }	{ “result”: [Answers] }
	2	{ }	{ “result”: “Insufficient input.” }
setNameQuestion()	1	{ “name”: “Question1” }	{ “result”: “Name is set” }

	2	{ }	{ "result": "Insufficient input." }
setAnswer()	1	{ "question_id": "1", "name": "Answer" }	{ "result": "Answer is set" }
	2	{ }	{ "result": "Insufficient input." }
getCorrect()	1	{ "question_id": "1" }	{ "result": "Correct answers" }
	2	{ }	{ "result": "Insufficient input." }
setCorrect()	1	{ "question_id": "1", "name": "Correct answer" }	{ "result": "Correct answer is set" }
	2	{ }	{ "result": "Insufficient input." }

Таблиця 14 – проектування тестових наборів методів програмних класів



## 4 Конструювання програмного продукту

### 4.1 Особливості конструювання структур даних

#### 4.1.1 Особливості інсталяції та роботи з СУБД

СУБД - MongoDB- система управління базами даних, що не вимагає опису схеми таблиць. Вважається одним з класичних прикладів NoSQL-систем, використовує JSON-подібні документи і схему бази даних. Застосовується в веб-розробці, зокрема, в рамках JavaScript.

Версія - MongoDB 4.4

MongoDB підходить для наступних застосувань:

- зберігання і реєстрація подій;
- системи управління документами і контентом;
- електронна комерція;
- дані моніторингу, датчиків;
- мобільні додатки;
- сховище операційних даних веб-сторінок (наприклад, зберігання коментарів, оцінок, профілів користувачів, сеанси користувачів).

#### 4.1.2 Особливості створення структур даних

Створення структур даних - JSON-структура - немає чітких вимог і стандартів, пов'язаних з його описом.

Нижче представлений код роботи структури даних:

```
const userControllr = require("../controllers/post_controller");
router.get("/posts", userControllr.findAll);
router.post("/add_post", userControllr.create);
router.get("/post/:id", userControllr.findOne);
router.put("/posts/:id", userControllr.UpdateUser);
router.delete("/posts/:id", userControllr.delete);
const Post = require("../models/post_model");

exports.create = (req, res) => {
  var db = req.db;
  var title = req.body.title;
  var description = req.body.description;
  var video = req.body.video;
```

```

var new_post = new Post({
  title: title,
  description: description,
  video: video
})
new_post.save(function (error) {
  if(error) {
    console.log(error)
  }
  res.send({
    success: true
  })
})
};

exports.findAll = (req, res) => {
  Post.find({}, 'title description video tests', function (error, posts) {
    if (error) {
      console.error(error);
    }
    res.send({
      posts: posts
    })
  }).sort({_id: -1})
};

exports.findOne = (req, res) => {
  var db = req.db;
  Post.findById(req.params.id, 'title description video tests', function (error, post) {
    if (error) {
      console.error(error);
    }
    res.send(post)
  })
};

```

## 4.2 Особливості конструювання програмних модулів

### 4.2.1 Особливості роботи з інтегрованим середовищем розробки

Середовище розробки – WebStorm.

WebStorm являє собою інструмент для розробки web-сайтів і редагування HTML, CSS і JavaScript коду. Рішення забезпечує швидку навігацію по файлах і генерує повідомлення про виникаючі проблеми в коді в режимі реального часу. JetBrains WebStorm дозволяє додавати розмітку HTML-документів або елементів SQL безпосередньо в JavaScript. JetBrains

					ІС КР 122 АІ-183 ПЗ	Л и
Змін	Л и	№.	П і д п	Д а		41

WebStorm здійснює розгортання і синхронізацію проектів через протокол FTP.

Використовуючи можливості коду HTML / XHTML і XML, WebStorm забезпечує автоматичне завершення стилів, посилань, атрибутів і інших елементів коду. При роботі з CSS здійснюється завершення коду класів, HTML-номерів, ключових слів і т. Д. WebStorm пропонує автоматичне рішення таких проблем, як вибір формату, властивостей, класів, посилань на файли і інших атрибутів CSS.

Рішення дозволяє використовувати потужність інструменту Zen coding для верстки HTML, відображає дії тега на web-сторінці. Продукт WebStorm здійснює завершення коду JavaScript для ключових слів, лейблів, змінних, параметрів і функцій DOM і підтримує специфічні особливості популярних браузерів. Реалізовані в рішенні функції рефакторінга JavaScript дозволяють перетворювати структуру коду і файлів і .js.

#### **4.2.2 Особливості створення програмної структури з урахуванням спеціалізованого Фреймворку**

Основні фреймворки:

Vue.js - це JavaScript бібліотека для створення веб-інтерфейсів з використанням шаблону архітектури MVVM (Model-View-ViewModel).

Функції Vue.js:

- Реактивні інтерфейси;
- Декларативний рендеринг;
- Зв'язування даних;
- Директиви (всі директиви мають префікс «V-». В директиву передається значення стану, а в якості аргументів використовуються html атрибути або Vue JS події);
- Логіка шаблонів;
- Компоненти;

- Обробка подій;
- Властивості;
- Переходи і анімація CSS;
- Фільтри.

2. Node.js - це платформа, заснована на середовищі виконання Chrome JavaScript, для простого створення швидких і масштабованих мережних додатків.

Node.js використовує керовану подіями неблокуючих модель введення / виведення, яка робить її легкою і ефективною, що ідеально підходить для додатків з інтенсивним використанням даних в реальному часі, що працюють на розподілених пристроях.

3. ExpressJS - це структура веб-додатків, яка надає вам простий API для створення веб-сайтів, веб-додатків і серверних частин. З ExpressJS вам не потрібно турбуватися про протоколах низького рівня, процесах і т. Д.

Express надає мінімальний інтерфейс для створення наших додатків. Він надає нам інструменти, необхідні для створення нашого застосування.

#### 4.2.3 Особливості створення програмних класів

Для більш зручної роботи з обіцянками існує спеціальний синтаксис, який називається «async / await». Він являє собою просте очікування Проміс, отримання значення і продовження функції. Функції, оголошені з використанням ключового слова async (асинхронні функції), дають нам можливість писати акуратний і не переобтяжений службовими конструкціями код, що дозволяє отримати той же результат, який ми отримували з використанням Проміс. Особливості створення показано на прикладі класу Course

```
class Course {
  title: {
    type: String,
    unique: true
  },
```

```

description: {
  type: String
},
video: {
  type: String
},
tests: [
  {
    question: {
      type: String
    },
    answerA: {
      type: String
    },
    answerB: {
      type: String
    },
    answerC: {
      type: String
    },
    answerD: {
      type: String
    },
    correct: {
      type: String
    }
  }
]
]
async addCourse () {
  await PostsService.addPost({
    title: this.title,
    description: this.description,
    video: this.video,
    tests: this.tests
  })
  this.$swal(
    'Great!',
    'Your post has been added!',
    'success'
  )
  await this.$router.push({ name: 'Posts' })
}
async Viewing () {
  const response = await PostsService.getData ()
  this.posts = response.data.posts
},
},
saveCourse (params) {
  return Api (). post ( 'add_post', params)
},

```

					IC KP 122 AI-183 ПЗ	Л и
Змін	Л и	№.	П і д п	Д а		44

```

getCourse () {
  return Api().get('posts')
}
}

```

#### 4.2.4 Особливості розробки алгоритмів методів програмних класів або процедур/функцій

Контролер забезпечує «зв'язок» між користувачем і системою, контролює і направляє дані від користувача до системи і навпаки, а також використовує модель і уявлення для реалізації необхідного дії. Нижче представлений контролер CourseController, який виконує весь функціонал роботи з курсами, а саме: створити курс, отримати усі курси, пройти обраний курс, оновити курс а також видалити його:

```

exports.create = (req, res) => {
  var db = req.db;
  var title = req.body.title;
  var description = req.body.description;
  var video = req.body.video;
  var new_post = new Post({
    title: title,
    description: description,
    video: video
  })
  new_post.save(function (error) {
    if(error) {
      console.log(error)
    }
    res.send({
      success: true
    })
  })
};

exports.findAll = (req, res) => {
  Post.find({}, 'title description video', function (error, posts) {
    if (error) {
      console.error(error);
    }
    res.send({
      posts: posts
    })
  }).sort({_id: -1})
};

exports.findOne = (req, res) => {

```

```

var db = req.db;
Post.findById(req.params.id, 'title description video', function (error, post) {
  if (error) {
    console.error(error);
  }
  res.send(post)
})
};

exports.delete = (req, res) => {
  var db = req.db;
  Post.remove({
    _id: req.params.id
  }, function (err, post) {
    if (err) {
      res.send(err)
    }
    res.send({
      success: true
    })
  })
};

exports.UpdateUser = (req, res) => {
  var db = req.db;
  Post.findById(req.params.id, 'title description video', function (error, post) {
    if (error) {
      console.error(error);
    }
    post.title = req.body.title
    post.description = req.body.description
    post.video = req.body.video
    post.save(function (error) {
      if (error) {
        console.log(error)
      }
      res.send({
        success: true
      })
    })
  })
});

```

### 4.3 Модульне тестування програмних класів

#### Перевірка функції Login

##### 1) Успішний вхід до аккаунту

					ІС КР 122 АІ-183 ПЗ	Л и
Змін	Л и	№.	П і д п	Д а		46



## Вход

yuri@gmail.com

\*\*\*\*\*

Войти

[Забыли пароль?](#)

[...или может](#)

[Просто нет аккаунта?](#)

Рисунок 4.3.1 – Вхідні дані

```
▼ {success: true, message: "The user is signed in to the account."} Login.vue?7463:76
  message: "The user is signed in to the account."
  success: true
  ► __proto__: Object
```

Рисунок 4.3.2 – Результат роботи тесту

## 2) Недостатня кількість параметрів

					ІС КР 122 АІ-183 ПЗ	Л и
						47
Змін	Л и	№.	П і д п	Д а		





# Вход

Войти

[Забыли пароль?](#)

[...или может](#)

Просто нет аккаунта?

Рисунок 4.3.3 – Вхідні дані

```
✖ POST http://localhost:8000/login 400 (Bad Request) xhr.js?b50d:177
Login.vue?7463:78
{success: false, message: "Failed to sign in to your account. Insufficient number of parameters."}
message: "Failed to sign in to your account. Insufficient number of parameters."
success: false
__proto__: Object
```

Рисунок 4.3.4 – Результат роботи тесту

### 3) Невірні дані



**Вход**

yuri@gmail.com

\*\*\*\*\*

**Войти**

Забыли пароль?

...или может

Просто нет аккаунта?

Рисунок 4.3.5 – Вхідні дані

```
✖ POST http://localhost:8000/login 400 (Bad Request) xhr.js?b50d:177
Login.vue?7463:78
{success: false, message: "Failed to sign in to your account. Login or password is not cor
rect."}
  message: "Failed to sign in to your account. Login or password is not correct."
  success: false
  __proto__: Object
```

Рисунок 4.3.6 – Результат роботи тесту

## 5 Розгортання та валідація програмного продукту

### 5.1 Інструкція з встановлення програмного продукту

Для нормальної роботи нового програмного продукту встановлення не потрібно. Необхідно мати браузер на телефоні, планшеті чи комп'ютері і за

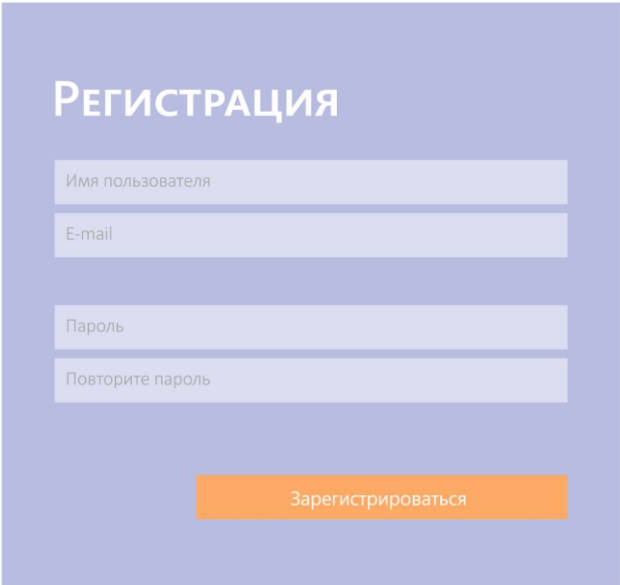
					ІС КР 122 АІ-183 ПЗ	Л и
						49
Змін	Л и	№.	П і д п	Д а		

допомогою маніпулятора «миша», якщо це комп'ютер чи сенсорного екрану зайти на сайт.

Установка серверної частини потрібна з боку розробників (тобто нашої команди). На сервері зберігається база даних і бекенд, який підключається до цієї бази даних. Для зв'язку клієнта з сервером використовується арі, за допомогою якого клієнт може здійснювати запити на сервер і отримувати від нього відповідь (наприклад, клієнт запитує у сервера список курсів, а сервер в свою чергу дістає цей список з бази даних та формує і відправляє відповідь клієнту) .

## 5.2 Інструкція з використання програмного продукту

ПП надає можливість користувачу ролі «гість» вести параметри реєстрації (імя користувача та його пароль), як показано на рисунку 5.1.



The image shows a registration form titled "РЕГИСТРАЦИЯ" (Registration) on a light blue background. Above the form is a circular logo with a stylized orange 't'. The form contains four input fields: "Имя пользователя" (Username), "E-mail", "Пароль" (Password), and "Повторите пароль" (Repeat password). Below these fields is an orange button labeled "Зарегистрироваться" (Register). At the bottom of the form, there is a link that says "...или может Просто есть аккаунт?" (or maybe there's just an account?).

Рис 5.1 - Пример экранной формы регистрации пользователя

Для реєстрації користувач повинен ввести значення, як показано на рисунку 5.2. При цьому необхідно пам'ятати про обмеження значення паролю та логіну.

					ІС КР 122 АІ-183 ПЗ	Л и
						50
Змін	Л и	№.	П і д п	Д а		



Рис 5.2 - Приклад екранної форми заповнення даних реєстрації користувача

### 5.3 Результати валідації програмного продукту

Метою програмного продукту є підвищення відсотка людей, які змогли знайти курс який їм підходить, щоб вивчити новий матеріал. Досягнення результату можна помітити на прикладі AL (AL - Access Level), описаного в пункті 1.2.1.2 (тобто кількості людей, які пройшли хоча б один курс / кількість користувачів сервісу). Результатом є отримана позитивна статистика, так як навіть як-що такого курсу не буде - то будь-який користувач дуже легко зможе виправити це - додавши його від свого імені, а отже дуже швидко найчастіші запитання користувачів буде задоволено. Ця статистика повідомляє нам, що сервіс допомагає людям (важливо: людям будь-якого віку, адже сервіс зроблений дуже простим і доступним всім).

					ІС КР 122 АІ-183 ПЗ	Л и
						51
Змін	Л и	№.	П і д п	Д а		

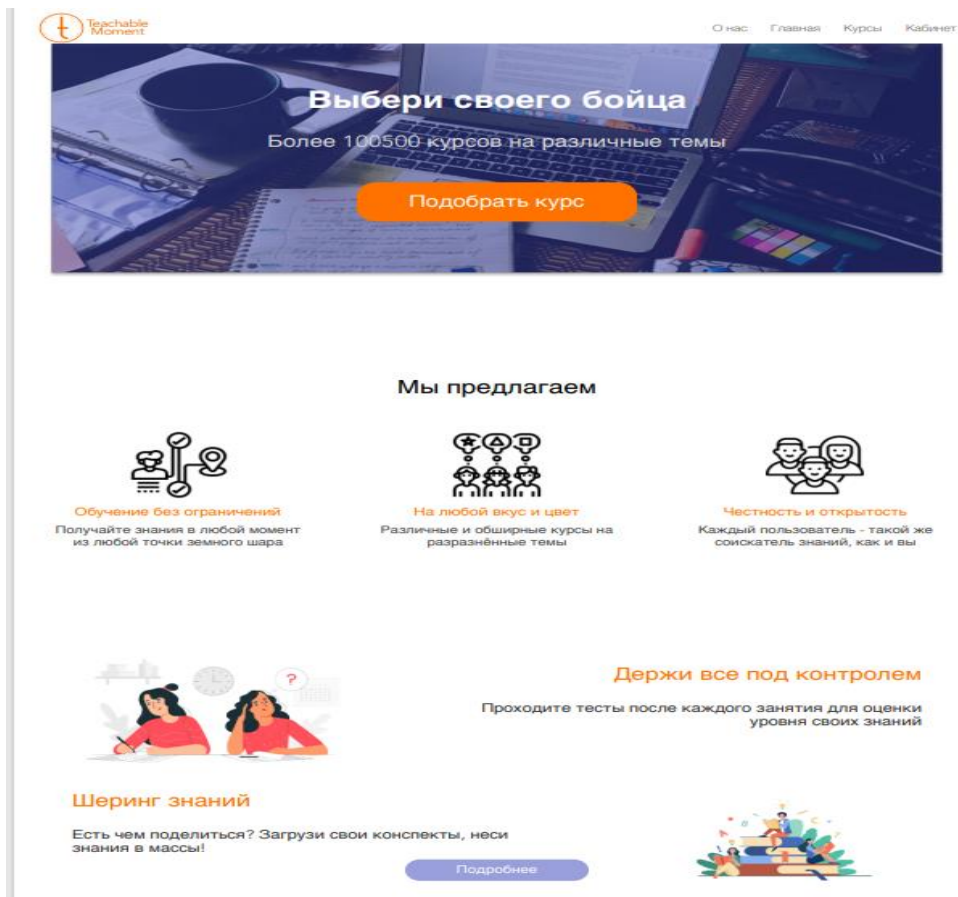


Рис 5.3 – результат виконаної роботи(головна сторінка)

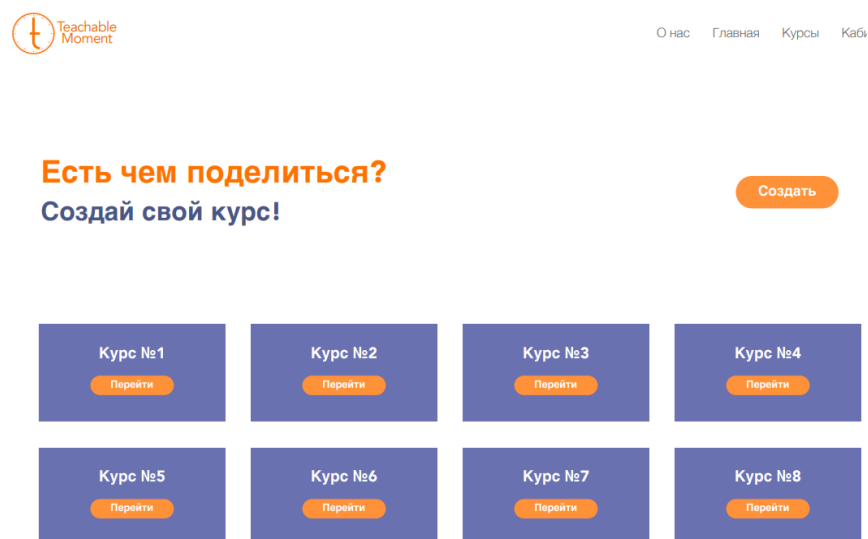
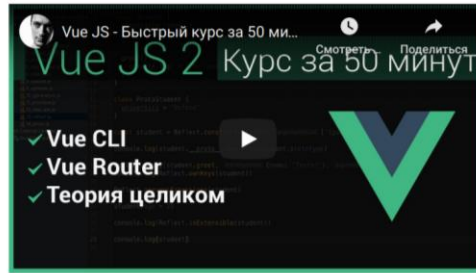


Рис 5.4 – результат виконаної роботи(створення власного курсу)

					ІС КР 122 АІ-183 ПЗ	Л и
						52
Змін	Л и	№.	П і д п	Д а		

## Курс №1



Vue.js (также Vue; /vjuː/) — JavaScript-фреймворк с открытым исходным кодом для создания пользовательских интерфейсов[4]. Легко интегрируется в проекты с использованием других JavaScript-библиотек. Может функционировать как веб-фреймворк для разработки одностраничных приложений в реактивном стиле.

Активация [Пройти тест](#)

Рис 5.5 – результат виконаної роботи( перегляд курсу)



[О нас](#) [Главная](#) [Курсы](#) [Кабинет](#)

## Тест к курсу №1

**Вопрос №1**

- ☐ Вариант ответа 1
- ☐ Вариант ответа 2
- ☐ Вариант ответа 3
- ☐ Вариант ответа 4

Рис 5.6 – результат виконаної роботи(проходження тестів)

					ІС КР 122 АІ-183 ПЗ	Л и
Змін	Л и	№.	П і д п	Д а		53

## ВИСНОВОК

В результаті створення програмного продукту була досягнута наступна мета його споживача: підвищення відсотка людей, які змогли знайти курс який їм підходить, щоб вивчити новий матеріал.

Доказом цього є наступні факти:

- 1) створено сайт- сервіс для навчання;
- 2) користувачі та гості можуть отримувати нові знання;
- 3) користувачі можуть ділитися своїми знаннями з іншими.

В процесі створення програмного продукту виникли такі труднощі (організаційні, проблеми відсутності досвіду, знань, потрібних в різних етапах):

- 1) проблема відсутності досвіду роботи в команді;
- 2) проблема відсутності потрібної кількості знань.

Через вищеописані непередбачені труднощі, а також через обмежений час на створення програмного продукту, команда реалізувала всі прецеденти та їх окремі кроки роботи.

Удосконалення та покращення планується реалізувати в майбутніх курсових роботах з урахуванням тем дисциплін наступних семестрів.

					ІС КР 122 АІ-183 ПЗ	Л и
						54
Змін	Л и	№.	П і д п	Д а		

## ПЕРЕЛІК ПОСИЛАНЬ

1. IT-блог «Хабрахабр». [Електронний ресурс] – Режим доступу: <https://habrahabr.ru/>
2. Документація бібліотеки Bootstrap [Електронний ресурс] – Режим доступу: <http://bootstrap-3.ru/index.php>
3. Документація JQuery [Електронний ресурс] – Режим доступу: <http://jquerybook.ru/api/>
4. Вікіпедія [Електронний ресурс] – Режим доступу: [https://ru.wikipedia.org/wiki/Заглавная\\_страница](https://ru.wikipedia.org/wiki/Заглавная_страница)
5. Dribbble [Електронний ресурс] – Режим доступу: <https://dribbble>.

					ІС КР 122 АІ-183 ПЗ	Л и
Змін	Л и	№.	П і д п	Д а		55