

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

## **КУРСОВА РОБОТА**

з дисципліни «Технології створення програмних продуктів»

за темою

"Розробка навчального WEB додатку для вивчення розділу геометрії «Вступ до стереометрії» з 3D інтерактивною візуалізацією просторових фігур"

Виконав(ла):  
студент 3-го курсу  
групи АІ-185  
Платоненко С.Д  
  
Перевірив:  
Блажко О. А.

Одеса-2020

### **Анотація**

В курсовій роботі розглядається процес створення програмного продукту – навчального WEB додатку.

Результати роботи розміщено на *github*-репозиторії за адресою:  
<https://github.com/Sergeev1ch/webproject>.

## **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

ПКМ – права клавіша миші

## **Зміст**

<b>1 Вимоги до програмного продукту.....</b>	<b>6</b>
<b>1.1 Визначення потреб споживача .....</b>	<b>6</b>
<b>1.1.1 Ієрархія потреб споживача.....</b>	<b>6</b>
<b>1.1.2 Деталізація матеріальної потреби.....</b>	<b>7</b>
<b>1.2 Бізнес-вимоги до програмного продукту .....</b>	<b>7</b>
<b>1.2.1 Опис проблеми споживача.....</b>	<b>7</b>
<b>1.2.1.1 Концептуальний опис проблеми споживача .....</b>	<b>7</b>
<b>1.2.1.2 Метричний опис проблеми споживача .....</b>	<b>7</b>
<b>1.2.2 Мета створення програмного продукту.....</b>	<b>8</b>
<b>1.2.2.1 Проблемний аналіз існуючих програмних продуктів.....</b>	<b>8</b>
<b>1.2.2.2 Мета створення програмного продукту.....</b>	<b>8</b>
<b>1.3 Вимоги користувача до програмного продукту .....</b>	<b>8</b>
<b>1.3.1 Історія користувача програмного продукту .....</b>	<b>8</b>
<b>1.3.2 Діаграма прецедентів програмного продукту .....</b>	<b>8</b>
<b>1.3.3 Сценаріїв використання прецедентів програмного продукту.....</b>	<b>9</b>
<b>1.4 Функціональні вимоги до програмного продукту .....</b>	<b>10</b>
<b>1.4.1. Багаторівнева класифікація функціональних вимог .....</b>	<b>10</b>
<b>1.4.2 Функціональний аналіз існуючих програмних продуктів .....</b>	<b>11</b>
<b>1.5 Нефункціональні вимоги до програмного продукту .....</b>	<b>12</b>
<b>1.5.1 Опис зовнішніх інтерфейсів.....</b>	<b>12</b>
<b>1.5.1.1 Опис інтерфейса користувача .....</b>	<b>12</b>
<b>1.5.1.1.1 Опис INPUT-інтерфейса користувача.....</b>	<b>12</b>
<b>1.5.1.1.2 Опис OUTPUT-інтерфейса користувача .....</b>	<b>12</b>
<b>1.5.1.2 Опис інтерфейсу із зовнішніми пристроями .....</b>	<b>13</b>
<b>1.5.1.3 Опис програмних інтерфейсів .....</b>	<b>13</b>
<b>1.5.1.4 Опис інтерфейсів передачі інформації .....</b>	<b>13</b>
<b>1.5.1.5 Опис атрибутів продуктивності .....</b>	<b>14</b>
<b>2 Планування процесу розробки програмного продукту .....</b>	<b>15</b>
<b>2.1 Планування ітерацій розробки програмного продукту .....</b>	<b>15</b>
<b>2.2 Концептуальний опис архітектури програмного продукту.....</b>	<b>15</b>

<b>2.3 План розробки програмного продукту .....</b>	<b>16</b>
<b>2.3.1 Оцінка трудомісткості розробки програмного продукту .....</b>	<b>16</b>
<b>2.3.2 Визначення дерева робіт з розробки програмного продукту .....</b>	<b>18</b>
<b>2.3.3 Графік робіт з розробки програмного продукту .....</b>	<b>18</b>
<b>2.3.3.1 Таблиця з графіком робіт .....</b>	<b>18</b>
<b>2.3.3.2 Діаграма Ганта.....</b>	<b>19</b>
<b>3 Проектування програмного продукту .....</b>	<b>20</b>
<b>3.1 Концептуальне та логічне проектування структур даних програмного продукту.....</b>	<b>20</b>
<b>3.1.1 Концептуальне проектування на основі UML-діаграми концептуальних класів.....</b>	<b>20</b>
<b>3.1.2 Логічне проектування структур даних .....</b>	<b>21</b>
<b>3.2 Проектування програмних класів .....</b>	<b>22</b>
<b>3.3 Проектування алгоритмів роботи методів програмних класів.....</b>	<b>22</b>
<b>3.4 Проектування тестових наборів методів програмних класів .....</b>	<b>24</b>
<b>4 Конструювання програмного продукту.....</b>	<b>28</b>
<b>4.1 Особливості конструювання структур даних .....</b>	<b>28</b>
<b>4.1.1 Особливості інсталяції та роботи з СУБД .....</b>	<b>28</b>
<b>4.1.2 Особливості створення структур даних .....</b>	<b>28</b>
<b>4.2 Особливості конструювання програмних модулів .....</b>	<b>30</b>
<b>4.2.1 Особливості роботи з інтегрованим середовищем розробки .....</b>	<b>30</b>
<b>4.2.2 Особливості створення програмної структури з урахуванням спеціалізованого Фреймворку.....</b>	<b>30</b>
<b>4.2.4 Особливості розробки алгоритмів методів програмних класів або процедур/функцій.....</b>	<b>31</b>
<b>4.3 Модульне тестування програмних класів .....</b>	<b>32</b>
<b>5 Розгортання та валідація програмного продукту .....</b>	<b>38</b>
<b>5.1 Інструкція з встановлення програмного продукту .....</b>	<b>38</b>
<b>5.2 Інструкція з використання програмного продукту .....</b>	<b>38</b>
<b>5.3 Результати валідації програмного продукту.....</b>	<b>40</b>
<b>Висновки.....</b>	<b>43</b>

## **1 Вимоги до програмного продукту**

### **1.1 Визначення 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.

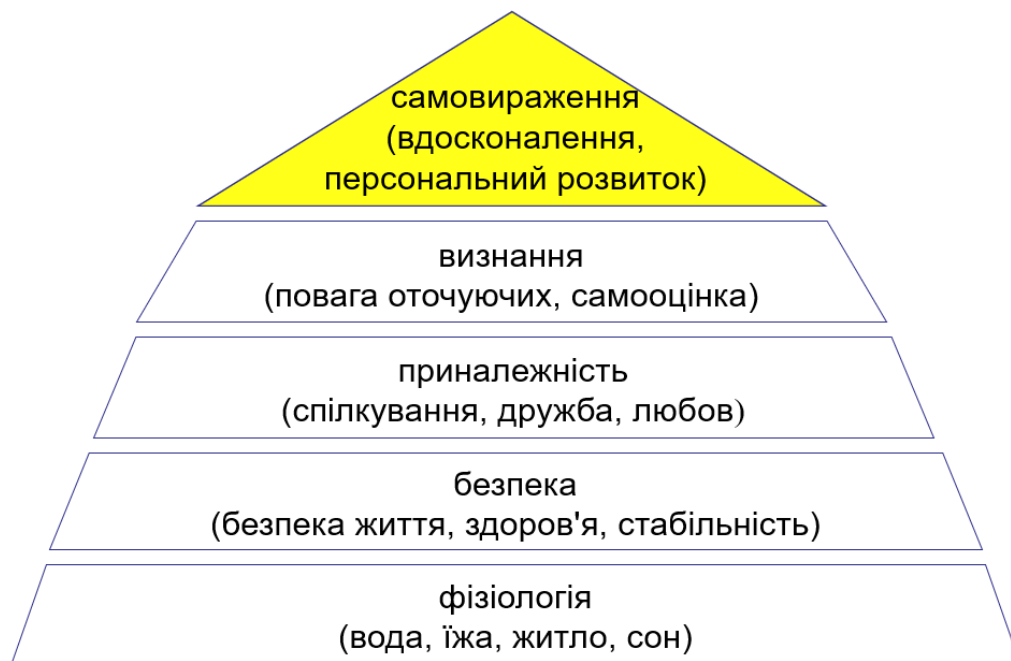


Рис. 1.1 – Приклад ієрархії потреби споживача

### 1.1.2 Деталізація матеріальної потреби

Для деталізації матеріальних потреб була розроблена MindMap, що представлена на рисунку 1.2.

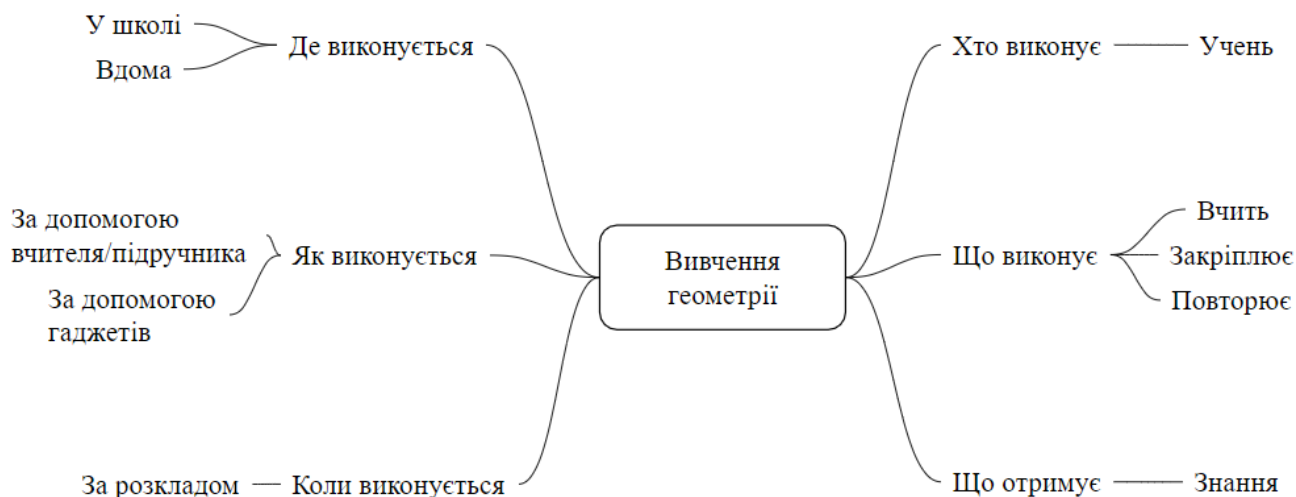


Рис. 1.2 – MindMap

## 1.2 Бізнес-вимоги до програмного продукту

### 1.2.1 Опис проблеми споживача

#### 1.2.1.1 Концептуальний опис проблеми споживача

1. Для кожної теми необхідно мінімум 3 приклади та 3 завдання.
2. Для кожного прикладу або завдання необхідно мінімум 1 графічне представлення.
3. Показник енергомичності розраховується за кількістю кліків необхідних для знаходження необхідної інформації.

#### 1.2.1.2 Метричний опис проблеми споживача

Таблиця 1.2.1.2 – Метричні показники проблем споживача

№	Загальний опис проблеми	Метричні показники
1	Відсутність прикладів / завдань	Недостатня кількість прикладів / завдань
2	Відсутність графічного представлення	Недостатня кількість графічних представлень
3	Важкий для сприйняття інтерфейс користувача	Низький показник енергомичності

## **1.2.2 Мета створення програмного продукту**

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

Таблиця 1.2.2.1 – Аналіз існуючих ПП

<b>№</b>	<b>Назва продукту</b>	<b>Вартість</b>	<b>Ступінь готовності</b>	<b>Примітка</b>
1	onlinemschool.com	безкоштовно	1	Відсутність прикладів, завдань
2	yaklass.ru	безкоштовно	1	Відсутнє графічне представлення
3	geogebra.org	безкоштовно	1	Важкий для сприйняття інтерфейс користувача

### **1.2.2.2 Мета створення програмного продукту**

Метою роботи є створення навчального посібника з зручним інтуїтивно-зрозумілим користувальницьким інтефейсом. Підвищення кількості прикладів та завдань для кожної теми, графічних представлень та показника енергомічності.

## **1.3 Вимоги користувача до програмного продукту**

### **1.3.1 Історія користувача програмного продукту**

Як користувач, я можу ознайомлюватися з теоретичним матеріалом

Як користувач, я можу отримувати завдання

Як користувач, я можу ознайомлюватися з розв'язанням задач

Як користувач, я можу взаємодіяти з об'єктами

### **1.3.2 Діаграма прецедентів програмного продукту**

Діаграма прецедентів зображена на рисунку 1.3.2.



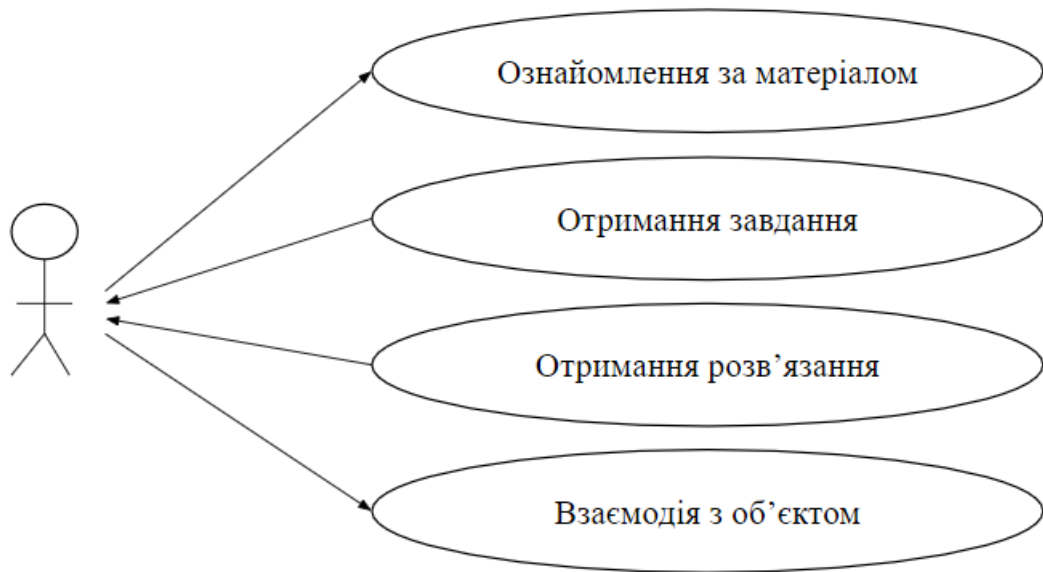


Рис. 1.3.2 – Діаграма прецедентів

### 1.3.3 Сценаріїв використання прецедентів програмного продукту

#### Сценарій №1.

Назва прецеденту : «Ознайомлення з матеріалом»

Передумова початку виконання сценарію : «Необхідність у вивченні, повторенні, закріпленні матеріалу»

Актор: «Користувач»

Гарантії успіху: «Вивчення, повторення, закріплення матеріалу»

Приклад успішного сценарію:

1. ПП надає можливість вибору теми
2. Користувач вибирає необхідну тему

#### Сценарій №2.

Назва прецеденту : «Отримання завдання»

Передумова початку виконання сценарію : «Ознайомлення з матеріалом»

Актор: «Користувач»

Гарантії успіху: «Вивчення, повторення, закріплення матеріалу»

Приклад успішного сценарію:

1. ПП відображає завдання
2. Користувач вибирає необхідне завдання

#### Сценарій №3.

Назва прецеденту : «Отримання розв'язання»

Передумова початку виконання сценарію : «Отримання завдання»

Актор: «Користувач»

Гарантії успіху: «Вивчення, повторення, закріплення матеріалу»

Приклад успішного сценарію:

1. Користувач відправляє запит на отримання розв'язання
2. ПП відображає результат запити

#### **Сценарій №4.**

Назва прецеденту : «Взаємодія з об'єктом»

Передумова початку виконання сценарію : «Ознайомлення з матеріалом»

Актор: «Користувач»

Гарантії успіху: «Вивчення, повторення, закріплення матеріалу»

Приклад успішного сценарію:

1. ПП відображає об'єкти можливі для взаємодії
2. Користувач вибирає об'єкт для взаємодії та надсилає запит
3. ПП відображає результат взаємодії користувача з об'єктом

### **1.4 Функціональні вимоги до програмного продукту**

#### **1.4.1. Багаторівнева класифікація функціональних вимог**

Багаторівнева класифікація функціональних вимог представлена в таблиці 1.4.1. На рисунку 1.4.1 опис ієрархічної класифікація функціональних вимог.

Таблиця 1.4.1 – Багаторівнева класифікація функціональних вимог

<b>Ідентифікатор функції</b>	<b>Назва функції</b>
FR1	Ознайомлення з матеріалом
FR1.1	ПП надає можливість вибору теми
FR1.2	Користувач обирає необхідну тему
FR2	Отримання завдання
FR2.1	ПП відображає завдання
FR2.2	Користувач обирає завдання
FR3	Отримання розв'язання
FR3.1	Користувач відправляє запит на отримання розв'язання

FR3.2	ППП відображає розв’язання
FR4	Взаємодія з об’єктами
FR4.1	ППП відображає об’єкти можливі для взаємодії
FR4.2	Користувач вибирає об’єкт для взаємодії та надсилає запит
FR4.3	ППП відображає результат взаємодії користувача з об’єктом

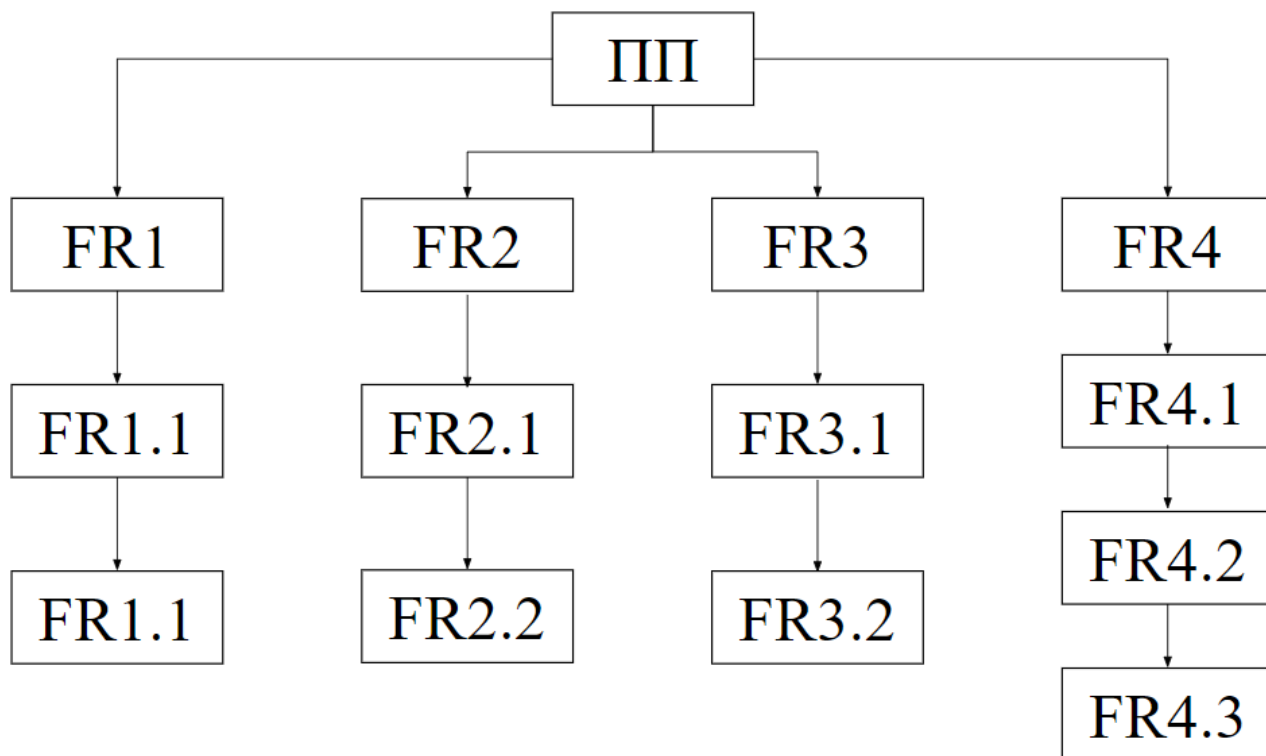


Рис. 1.4.1 – Опис ієрархічної класифікація функціональних вимог

#### 1.4.2 Функціональний аналіз існуючих програмних продуктів

Таблиця 1.4.2 – Функціональний аналіз існуючих програмних продуктів

Ідентифікатор функції	onlinemschool.com	yaklass.ru	geogebra.org
FR1	+	+	+
FR2	+	+	-
FR3	-	-	-
FR4	-	-	+

## 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

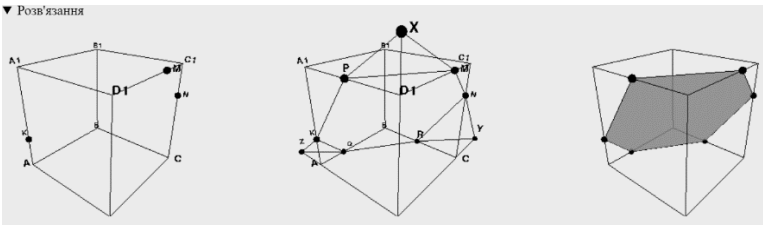
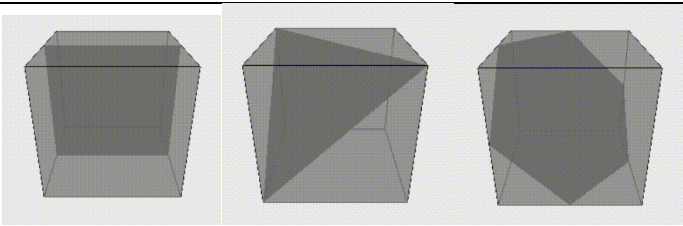
Таблиця 1.5.1.1.1 – Опис INPUT-інтерфейса користувача

Ідентифікатор функції	Засіб INPUT потоку	Особливості використання
FR1	2/3-кнопочний маніпулятор типу "миша"	Використовуючи меню здійснюється навігація
FR2	2/3-кнопочний маніпулятор типу "миша"	
FR3	2/3-кнопочний маніпулятор типу "миша"	Для отримання роз'язання необхідно використовувати сполер
FR4	2/3-кнопочний маніпулятор типу "миша"	Використання лівої кнопки миші для взаємодії з об'єктами

##### 1.5.1.1.2 Опис OUTPUT-інтерфейса користувача

Таблиця 1.5.1.1.2 – Опис OUTPUT-інтерфейса користувача

Ідентифікатор функції	Засіб OUTPUT потоку	Особливості використання
FR1.1	Графічний інтерфейс	

FR2.2	Графічний інтерфейс	<b>Завдання 1</b> Дано призму $ABCD A_1 B_1 C_1 D_1$ . Побудувати переріз призми площиною, що проходить через точки: $M \in A_1 B_1 C_1 D_1$ , $A$ і $D$ . ► Розв'язання
FR3.2	Графічний інтерфейс	▼ Розв'язання 
FR4.3	Графічний інтерфейс	

### 1.5.1.2 Опис інтерфейсу із зовнішніми пристроями

Таблиця 1.5.1.2 – Опис інтерфейсу із зовнішніми пристроями

Ідентифікатор функції	Зовнішній пристрій
FR1	Комп'ютер, ноутбук
FR2	Комп'ютер, ноутбук
FR3	Комп'ютер, ноутбук
FR4	Комп'ютер, ноутбук

### 1.5.1.3 Опис програмних інтерфейсів

Для реалізації більшості функцій програмного продукту необхідно:

- HTML
- CSS
- JavaScript
- jQuery
- Three.js

### 1.5.1.4 Опис інтерфейсів передачі інформації

Рекомендовано використовувати:

- Ethernet
- WiFi

### 1.5.1.5 Опис атрибутів продуктивності

Ідентифікатор функції	Максимальний час реакції програмного продукту на дії користувача(секунди)
FR1	4
FR2	4
FR3	4
FR4	4

## 2 Планування процесу розробки програмного продукту

### 2.1 Планування ітерацій розробки програмного продукту

З метою забезпечення для вимог таких рекомендацій IEEE-стандарту, як необхідність, корисність при експлуатації, здійсненність функціональних вимог до ПП, визначте функціональні пріоритети, які будуть використані при плануванні ітерацій розробки ПП.

Таблиця 2.1 – Опис функціональних пріоритетів

Ідентифікатор функції	Функціональні залежності	Вплив на досягнення мети, %	Пріоритет функції
FR1	-		M
FR2	FR1		M
FR3	FR1		M
FR4	FR1		S

### 2.2 Концептуальний опис архітектури програмного продукту

Архітектура ПП представлена на рисунку 2.2.

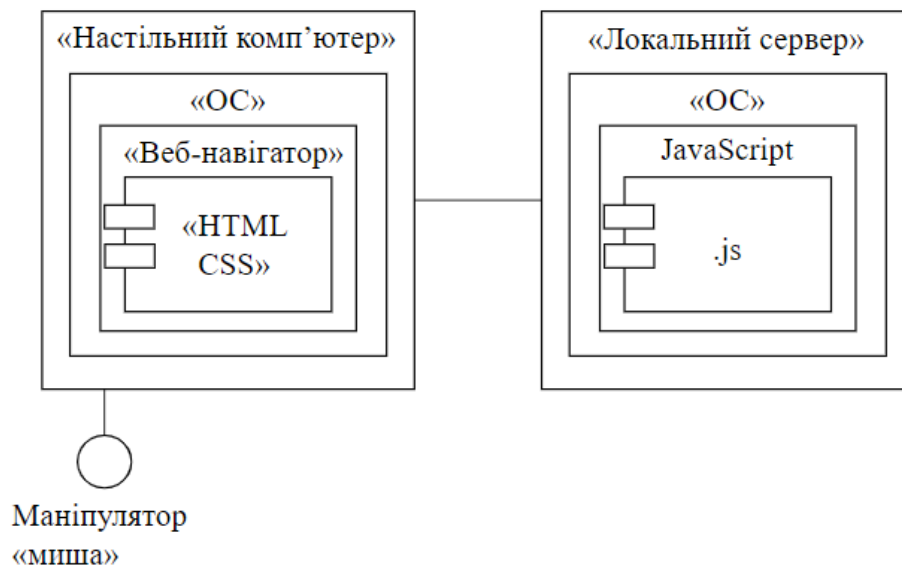


Рисунок 2.2 – Архітектура програмного продукту

## 2.3 План розробки програмного продукту

### 2.3.1 Оцінка трудомісткості розробки програмного продукту

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 (рис. 2.3.1.1).

Тип актора	Ваговий коефіцієнт
Простий	1
Середній	2
Складний	3

Рис. 2.3.1.1 – Вагові коефіцієнти акторів

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 (рис. 2.3.1.2).

Тип прецедента	Кількість кроків сценарію	Ваговий коефіцієнт
Простий	$\leq 3$	5
Середній	4-7	10
Складний	$> 7$	15

Рис. 2.3.1.2 – Вагові коефіцієнти прецедентів

$$UCPP = 1 * 1 + 4 * 5 = 21$$

Технічна складність проекту (TCF – Technical Complexity Factor) обчислюється з урахуванням показників технічної складності (табл. 2.3.1.1). Кожному показнику присвоюється значення STi в діапазоні від 0 до 5: 0 означає



відсутність значимості показника для даного проекту, 5 - високу значимість).  
Значення TCF обчислюється за формулою –  $TCF = 0,6 + (0,01 * (ST_i * Вага_i))$

Таблиця 2.3.1.1 – TCF-таблиця

Показник	Опис показника	Вага
T1	Розподільна система	2
T2	Висока продуктивність	1
T3	Робота користувачів онлайн	1
T4	Складна обробка даних	-1
T5	Повторне використання коду	1
T6	Простота інсталювання	0.5
T7	Простота використання	0.5
T8	Переносимість	2
T9	Простота внесення змін	1
T10	Паралелізм	1
T11	Спеціальні вимоги безпеки	1
T12	Беспосередній доступ до системи зі сторони зовнішніх користувачів	1
T13	Спеціальні вимоги до навчання користувачів	1

$TCF = 0.81$

Рівень кваліфікації розробників (EF - Environmental Factor) обчислюється з урахуванням наступних показників (табл. 2.3.1.2).

Таблиця 2.3.1.2 – Рівень кваліфікації розробників

Показник	Опис показника	Вага
F1	Знайомство з технологією	1.5
F2	Досвід розробки додатків	0.5
F3	Досвід використання ООП	1
F4	Наявність провідного аналітика	0.5
F5	Мотивація	1
F6	Стабільність вимог	2
F7	Часткова зайнятість	-1

F8	Складні мови програмування	-1
----	----------------------------	----

EF = 0.98

UCP = 16.66

### 2.3.2 Визначення дерева робіт з розробки програмного продукту

При створенні дерева робіт (Work BreakDown Structure- WBS) використовується дерево функцій.

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 Sub Task (WST) з урахуванням базових процесів розробки програмних модулів: проектування, конструювання, модульне тестування, збірка та системне тестування(рис. 2.3.2).

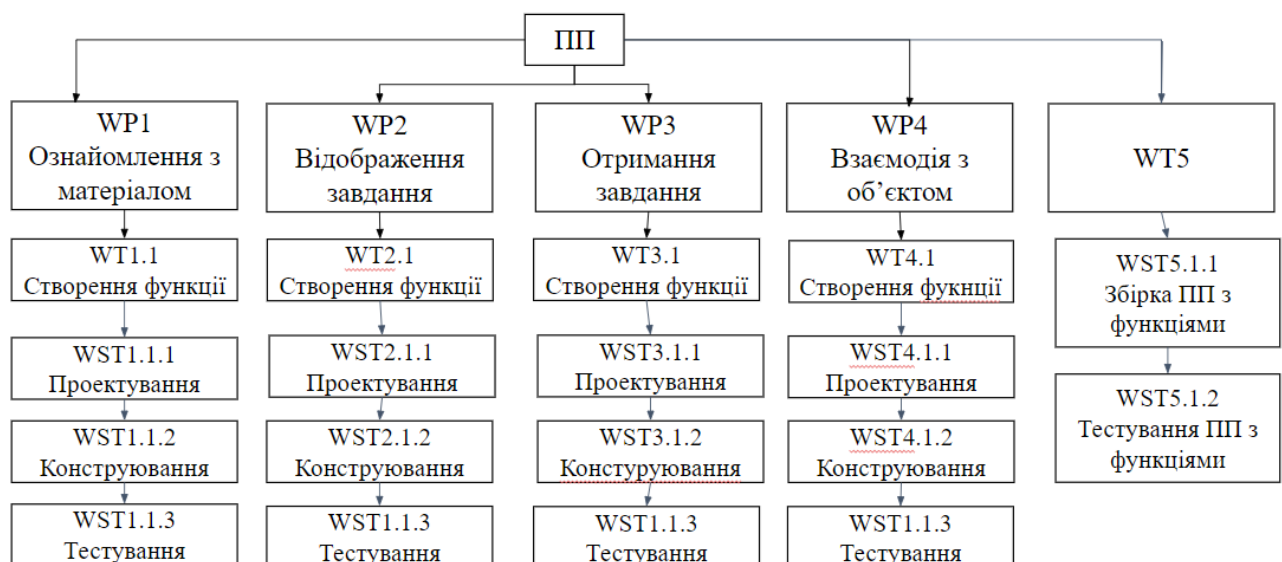


Рис. 2.3.2 – Дерево робіт з розробки ПП

### 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 2.3.3.1.

Таблиця 2.3.3.1 – Таблиця графіку робіт

WST	Дата початку	Дні	Дата завершення	Виконавець
1.1.1	18.10.20	2	20.10.20	Платоненко
1.1.2	20.10.20	2	22.10.20	Платоненко
1.1.3	22.10.20	2	24.10.20	Платоненко
2.1.1	24.10.20	2	26.10.20	Платоненко

2.1.2	26.10.20	2	28.10.20	Платоненко
2.1.3	28.10.20	3	31.10.20	Платоненко
3.1.1	31.10.20	3	3.11.20	Платоненко
3.1.2	3.11.20	3	6.11.20	Платоненко
3.1.3	6.11.20	3	9.11.20	Платоненко
4.1.1	9.11.20	2	11.11.20	Платоненко
4.1.2	11.11.20	3	14.11.20	Платоненко
4.1.3	14.11.20	3	17.11.20	Платоненко
5.1.1	17.11.20	5	22.11.20	Платоненко
5.1.2	22.11.20	5	27.11.20	Платоненко

### 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(рис.2.3.3.2).

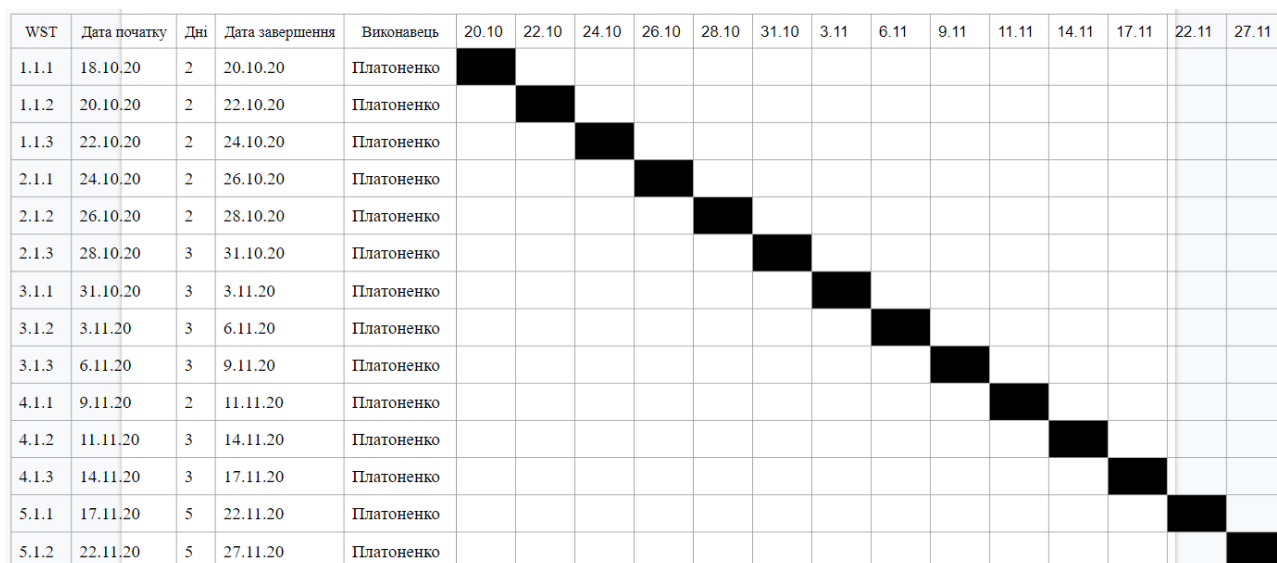


Рис. 2.3.3.2 – Діаграма Ганта

### 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

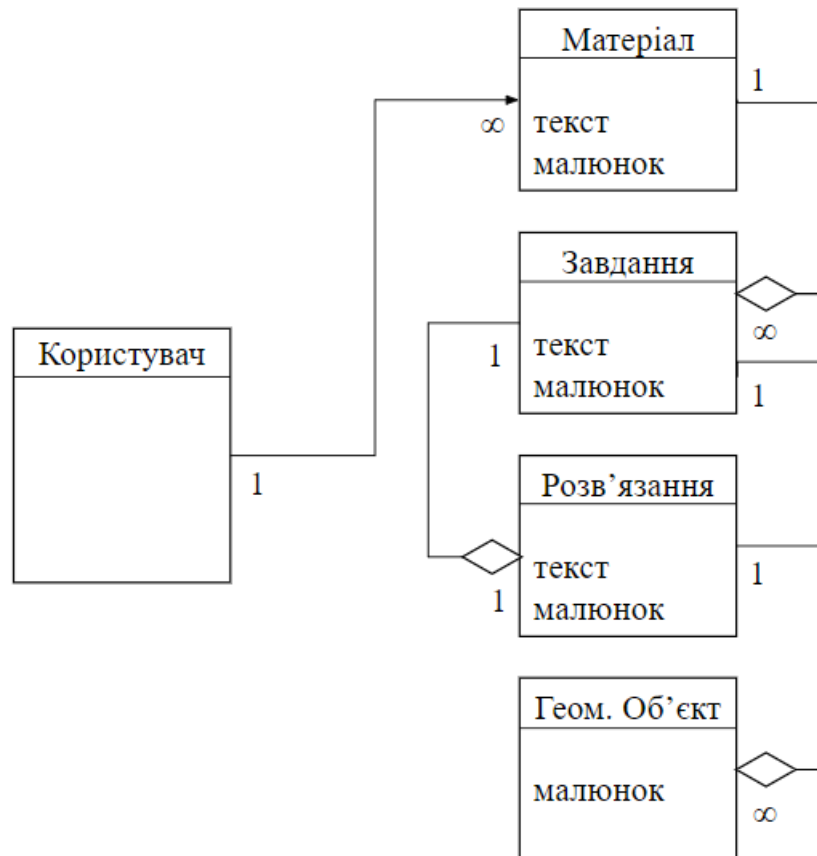


Рис. 3.1.1 – UML-діаграма концептуальних класів

### 3.1.2 Логічне проектування структур даних

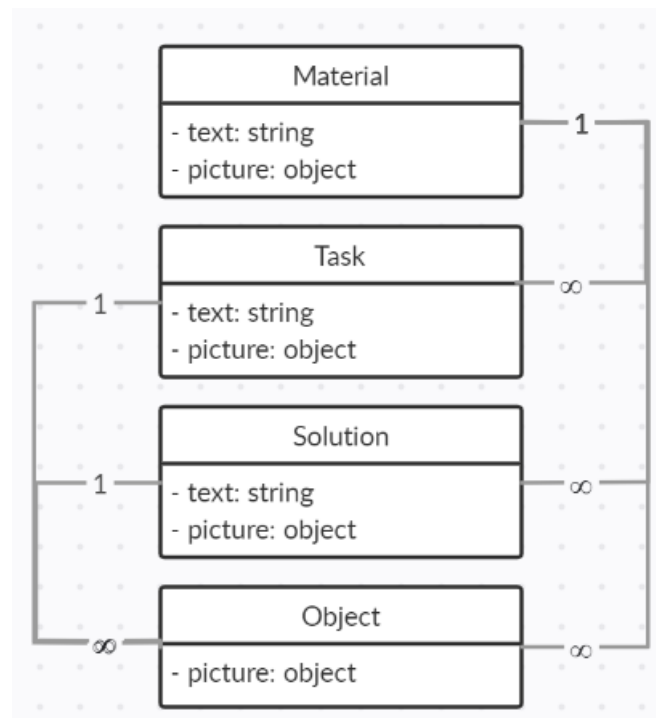


Рис. 3.1.2 – UML-діаграма структурних класів

### 3.2 Проектування програмних класів

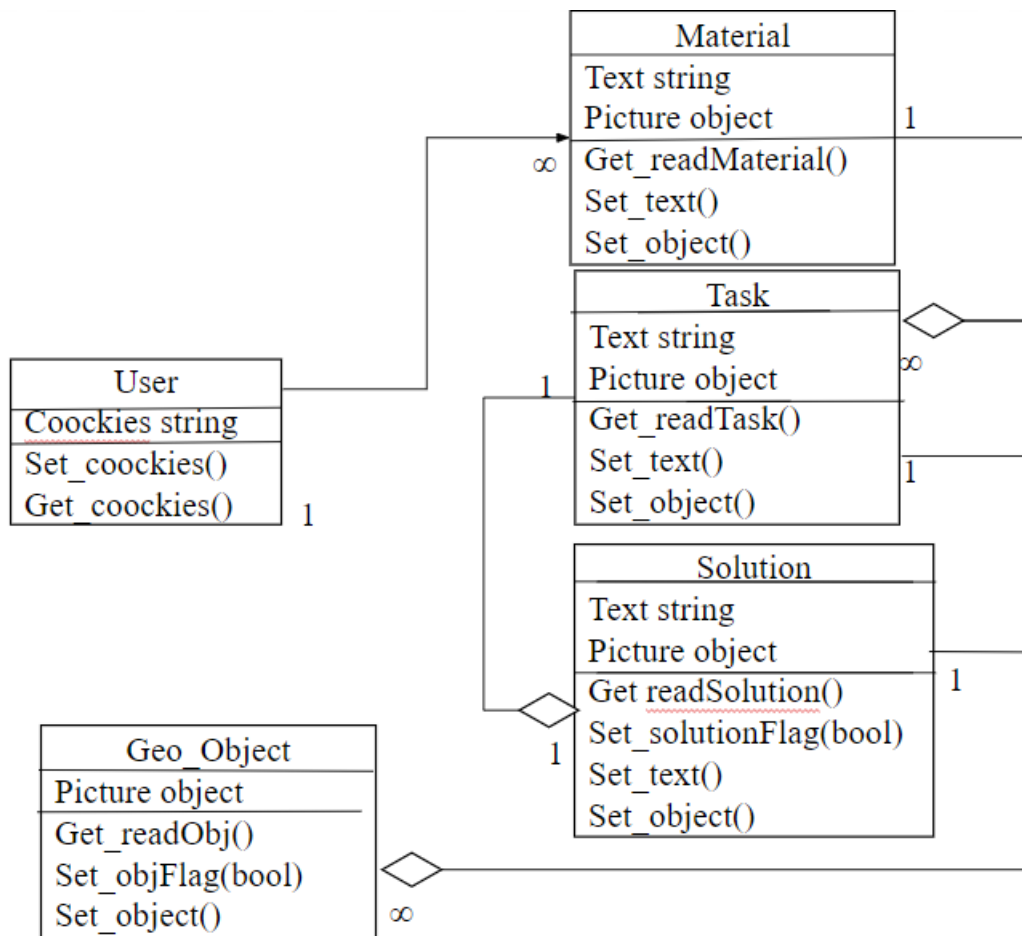


Рис. 3.2 – Діаграма програмних класів

### 3.3 Проектування алгоритмів роботи методів програмних класів

Алгоритм класу Get\_readMaterial зображений на рисунку 3.3.1.

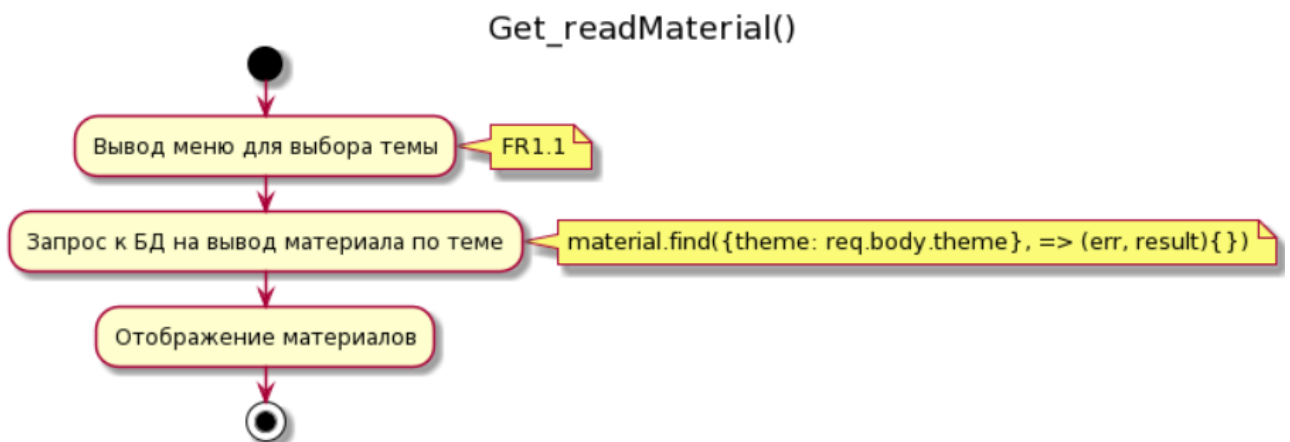


Рисунок 3.3.1 – Алгоритм класу Get\_readMaterial

```

start
title Get_readMaterial()
:Вывод меню для выбора темы;
  note right
  FR1.1
  end note
:Запрос к БД на вывод материала по теме;
  note right
  material.find({theme: req.body.theme}, => (err, result){ })
  end note
:Отображение материалов;
stop
@enduml

```

Алгоритм класу Get\_readTask зображений на рисунку 3.3.2.

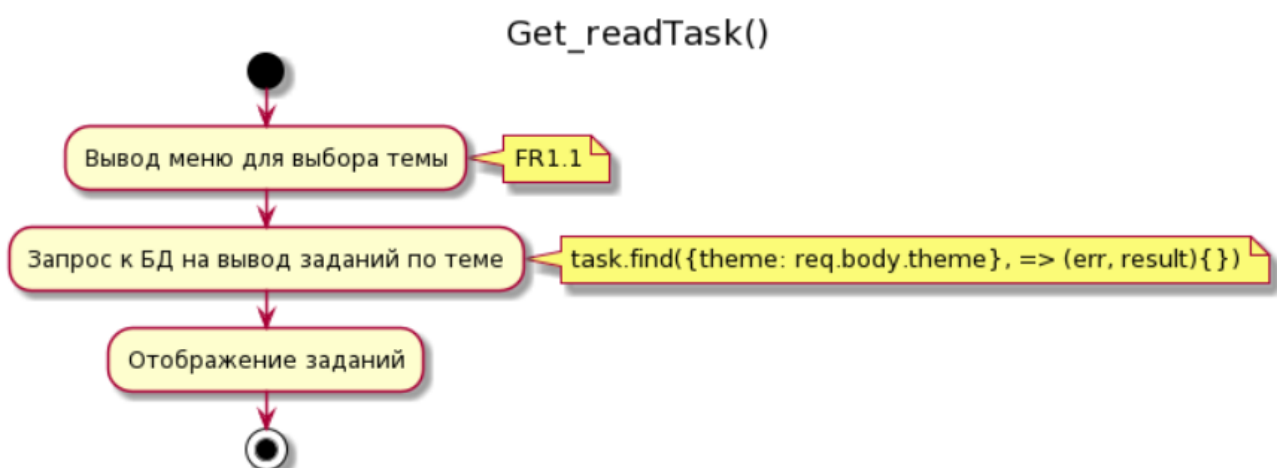


Рисунок 3.3.2 – Алгоритм класу Get\_readTask

Алгоритм класу Get\_readSolution зображений на рисунку 3.3.3.

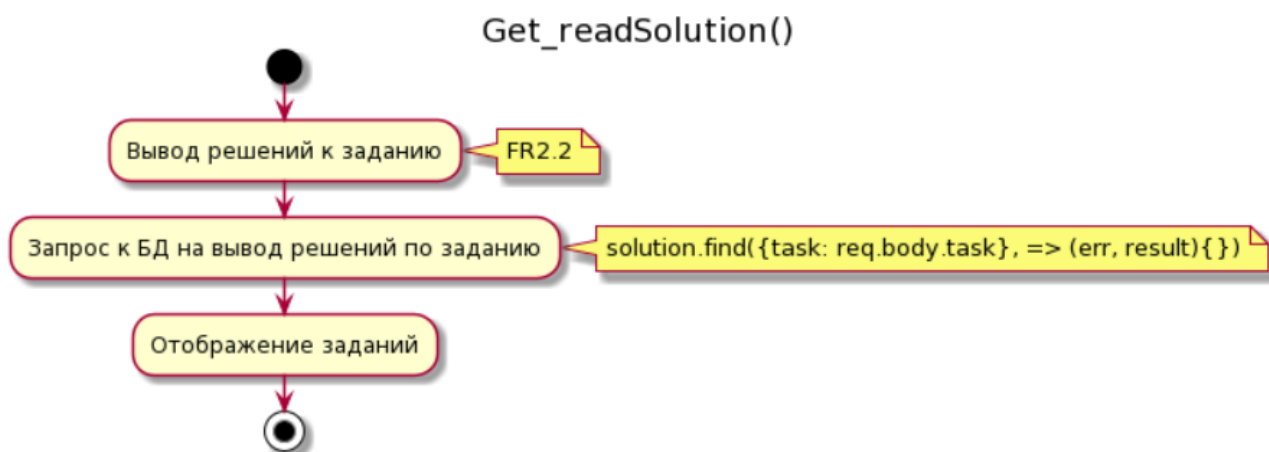


Рисунок 3.3.3 – Алгоритм класу Get\_readSolution

Алгоритм класу Get\_readObj зображений на рисунку 3.3.4.

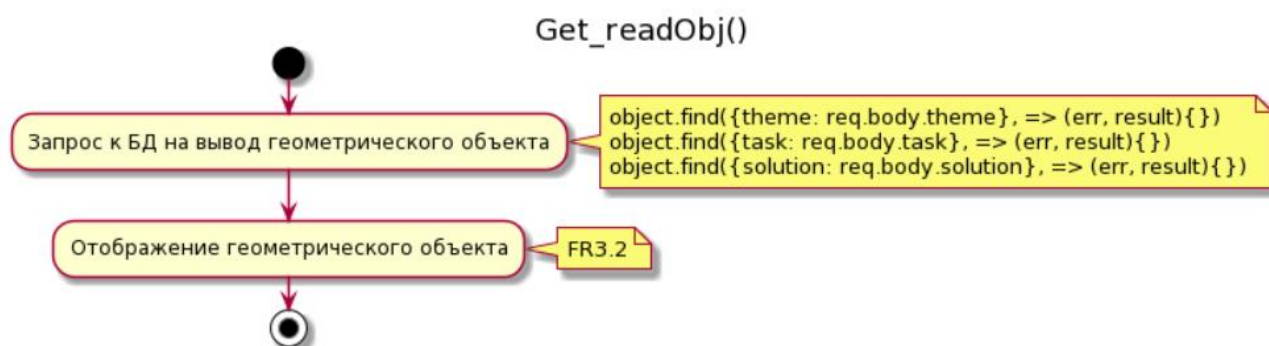


Рисунок 3.3.4 – Алгоритм класу Get\_readObj

### 3.4 Проектування тестових наборів методів програмних класів

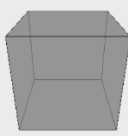
Таблиця 3.4.1 – Тестування функції get\_menu.

Назва функції	№ тесту	Опис вхідних значень	Опис очікуваних результатів
get_menu	1		Мокап форма



			<p><b>ВСТУП ДО СТЕРЕОМЕТРІЇ</b></p> <p>Основні поняття стереометрії. Аксіоми стереометрії</p> <p>Наслідки з аксіом стереометрії</p> <p>Просторові фігури. Початкові відомості про многогранники</p> <p><b>Призма</b></p> <p>Трикутна призма</p> <p>Чотирикутна призма</p> <p>П'ятикутна призма</p> <p><b>Піраміда</b></p> <p>Трикутна піраміда</p> <p>Чотирикутна піраміда</p> <p>П'ятикутна піраміда</p> <p><b>Конус</b></p> <p><b>Циліндр</b></p>
--	--	--	---

Таблиця 3.4.2 – Тестування функції get\_material.

Назва функції	№ тесту	Опис вхідних значень	Опис очікуваних результатів
get_material	1		<p><b>Мокап форма</b></p> <p><b>ЧОТИРИКУТНА ПРИЗМА</b></p>  <p><b>Чотирикутна призма (куб)</b> - многогранник, всі грані якого - квадрати.</p> <p><b>Грань куба</b> - це частина площини, обмежена сторонами квадрата.</p> <ul style="list-style-type: none"> <li>- куб має вісім граней;</li> <li>- кожна грань куба перетинається з чотирма іншими гранями під прямим кутом і паралельно протилежній межі;</li> <li>- межі мають однакову площу, яку можна знати, використавши формули для обчислення площі квадрата.</li> </ul> <p><b>Рібро куба</b> - це відрізок, утворений перетином двох граней куба.</p> <ul style="list-style-type: none"> <li>- куб має дванадцять ребер;</li> <li>- кожен кінець ребра з'єднаний з двома сусідніми ребрами під прямим кутом;</li> <li>- ребра куба мають однакову довжину.</li> </ul> <p><b>Вершина куба</b> - це найвіддаленіша від центру куба точка, яка лежить на перетині трьох граней куба.</p> <ul style="list-style-type: none"> <li>- куб має вісім вершин;</li> <li>- кожна вершина утворена тітськи трьома гранями і трьома ребрами.</li> </ul>

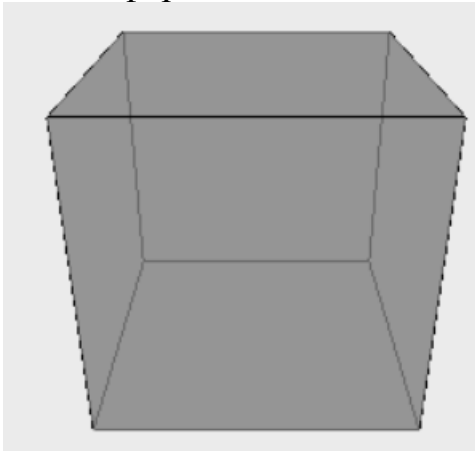
Таблиця 3.4.3 – Тестування функції get\_task.

Назва функції	№ тесту	Опис вхідних значень	Опис очікуваних результатів
get_task	1		<p><b>Мокап форма</b></p> <p><b>Завдання 1</b> Дано призму <math>ABCA_1B_1C_1D_1</math>. Побудувати переріз призми площиною, що проходить через точки: <math>M \in A_1B_1C_1D_1</math>, <math>A</math> і <math>D</math>.</p>

Таблиця 3.4.4 – Тестування функції get\_solution.

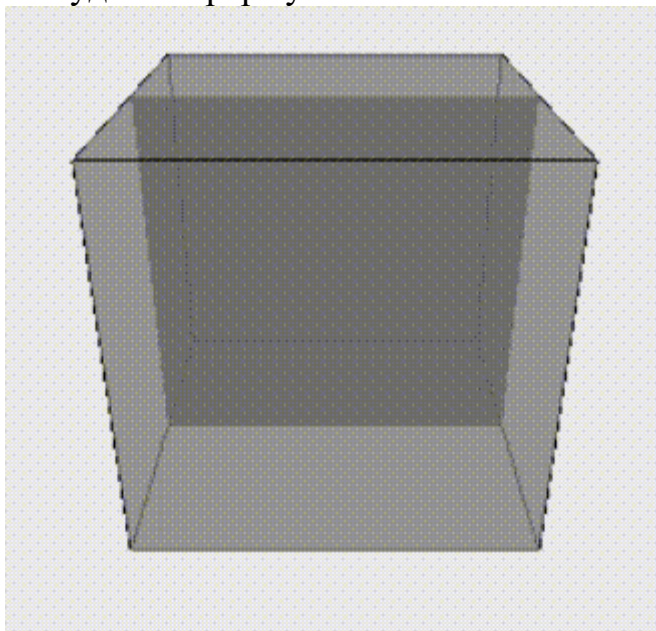
Назва функції	№ тесту	Опис вхідних значень	Опис очікуваних результатів
get_solution	1		Мокап форма 

Таблиця 3.4.5 – Тестування функції get\_object.

Назва функції	№ тесту	Опис вхідних значень	Опис очікуваних результатів
get_object	1		Мокап форма 

Таблиця 3.4.6 – Тестування функції use\_object.

Назва функції	№ тесту	Опис вхідних значень	Опис очікуваних результатів
use_object	1		Обертання об'єкта навколо осі
use_object	2		Завершення обертання

use_object	3	Мокап форма Побудова перерізу	
------------	---	----------------------------------	--

## 4 Конструювання програмного продукту

### 4.1 Особливості конструювання структур даних

#### 4.1.1 Особливості інсталяції та роботи з СУБД

MongoDB реалізує новий підхід до побудови баз даних, де немає таблиць, схем, запитів SQL, зовнішніх ключів і багатьох інших речей, які притаманні об'єктно-реляційним базам даних. Одним з популярних стандартів обміну даними та їх зберігання є JSON (JavaScript Object Notation). JSON ефективно описує складні за структурою дані. Спосіб зберігання даних в MongoDB в цьому плані схожий на JSON, хоча формально JSON не використовується. Для зберігання в MongoDB застосовується формат, який називається BSON (Бісон) або скорочення від binary JSON.

Для інсталяції СУБД MongoDB необхідно перейти за посилання <https://www.mongodb.com/try/download/community>.  
Обрати необхідну версію та завантажити виконуючий файл.

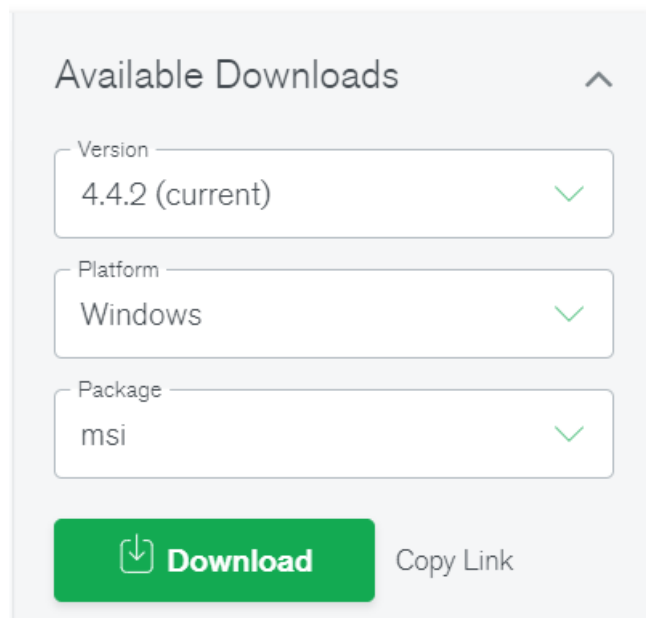


Рис. 4.1.1 – Вибір виконуючого файлу

#### 4.1.2 Особливості створення структур даних

Для створення бази даних, та її об'єктів використовується графічний інтерфейс.

Create Database

Database Name

Collection Name

☐ Capped Collection ⓘ
   
☐ Use Custom Collation ⓘ

Before MongoDB can save your new database, a collection name must also be specified at the time of creation. [More Information](#)

CANCEL

CREATE DATABASE

Рис. 4.1.2.1 – Створення БД

Create Collection

Collection Name

☐ Capped Collection ⓘ
   
☐ Use Custom Collation ⓘ

CANCEL

CREATE COLLECTION

Рис. 4.1.2.2 – Створення колекції

Insert to Collection bank.personal

VIEW {} ≡

```

1 ▾ /**
2   * Paste one or more documents here
3   */
4 ▾ {
5   |   "_id": {
6   |     "$oid": "5fc40bb22f308f9f5c99388b"
7   |   }
8   }

```

CANCEL

INSERT

Рис. 4.1.2.3 – Створення об'єкту БД

## 4.2 Особливості конструювання програмних модулів

### 4.2.1 Особливості роботи з інтегрованим середовищем розробки

Sublime Text – потужний текстовий редактор коду. Він швидко обробить просту веб-сторінку або програму на сто тисяч рядків коду. Редактор містить різні візуальні теми, з можливістю завантаження додаткових. Користувачі бачать весь свій код в правій частині екрану у вигляді міні-карти, при кліці на яку можна здійснювати навігацію. Є кілька режимів екрану. Один з них включає від 1 до 4 панелей, за допомогою яких можна показувати до чотирьох файлів одночасно.

Повноцінний (free modes) режим показує тільки один файл без будь-яких додаткових навколо нього меню. Виділення стовпців цілком або розстановка кількох покажчиків по тексту, що робить можливим миттєву правку. Команди типу: переміщення на знак, переміщення на рядок, вибірка тексту, переміщення на слово або його частини (CamelCase, розділений дефісом або підкресленням), перехід на початок або кінець рядка тощо, Впливає на всі покажчики незалежно і відразу, дозволяючи правити складно структурований текст швидко, без використання макрокоманд або регулярних виразів.

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

Для візуалізацій геометричних об'єктів була використана бібліотека Three.js  
Three.js – легка кросбраузерна бібліотека на мові JavaScript для роботи з тривимірною анімованою графікою.

Технологія WebGL дозволяє вбудовувати в сайт тривимірну графіку, яка використовує графічний прискорювач пристрою без використання будь-яких плагінів для браузера.

Приклад створення об'єкту за допомогою Three.js

```
var scene = new THREE.Scene();
var camera = new THREE.PerspectiveCamera( 75, window.innerWidth /
window.innerHeight, 0.1, 1000 );
var renderer = new THREE.WebGLRenderer();
renderer.setSize( window.innerWidth, window.innerHeight );
document.body.appendChild( renderer.domElement );
var geometry = new THREE.BoxGeometry( 10, 10, 10);
var material = new THREE.MeshBasicMaterial( { color: 'grey' } );
var cube = new THREE.Mesh( geometry, material );
scene.add( cube );
camera.position.z = 25;
```

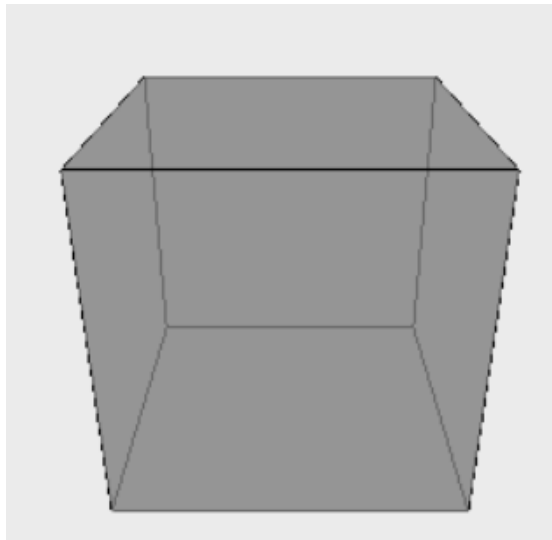


Рис. 4.2.2 – Об'єкт створений за допомогою Three.js

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

##### Get\_readMaterial

```
app.get('/theme:id', (req, res) => {
    material.find({theme.id}, (err, allMaterial) => { // Запит до БД
        if (err) {
            res.render('theme', { data: null, message: req.flash('err') }) // Виведення помилки,
якщо запит неуспішний
        } else {
            res.render('theme', { data: allMaterial }) // Відправлення результатів запиту для
відображення
        }
    })
})
```

##### Get\_readTask

```
app.get('/theme:id', (req, res) => {
    task.find({theme.id}, (err, allTask) => { // Запит до БД
        if (err) {
            res.render('theme', { data: null, message: req.flash('err') }) // Виведення помилки,
якщо запит неуспішний
        } else {
            res.render('theme', { data: allTask }) // Відправлення результатів запиту для
відображення
        }
    })
})
```

##### Get\_readSolution

```
app.get('/theme:id', (req, res) => {
```

```

        solution.find({task.id}, (err, allSolution) => { // Запит до БД
        if (err) {
            res.render('theme', { data: null, message: req.flash('err') }) // Виведення помилки,
якщо запит неуспішний
        } else {
            res.render('theme', { data: allSolution }) // Відправлення результатів запиту для
відображення
        }
        })
    })

```

### Get\_readObj

```

app.get('/theme:id', (req, res) => {
    object.find({theme.id}, (err, allObject) => { // Запит до БД
    if (err) {
        res.render('theme', { data: null, message: req.flash('err') })// Виведення помилки,
якщо запит неуспішний
    } else {
        res.render('theme', { data: allObject }) / Відправлення результатів запиту для
відображення
    }
    })
})

```

## 4.3 Модульне тестування програмних класів

get_menu	1		мокап форма
----------	---	--	-------------

```

app.get('/theme:id', (req, res) => {
    themes.find({}, (err, allTheme) => { // Запит до БД
    if (err) {
        res.render('theme', { data: null, message: req.flash('err') })
    } else {
        res.render('theme', { data: allTheme }) // Відправлення результатів запиту для
відображення
    }
    })
})

```



ВСТУП ДО СТЕРЕОМЕТРІЇ
Основні поняття стереометрії. Аксіоми стереометрії
Наслідки з аксіом стереометрії
Просторові фігури. Початкові відомості про многогранники
Призма
Трикутна призма
Чотирикутна призма
П'ятикутна призма
Піраміда
Трикутна піраміда
Чотирикутна піраміда
П'ятикутна піраміда
Конус
Циліндр

Рис. 4.3.1 – Результат тестування функції з відображення меню

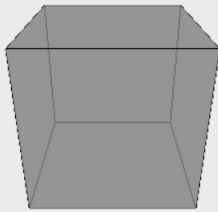
get_material	2		мокап форма
--------------	---	--	-------------

```

app.get('/theme:id', (req, res) => {
  material.find({theme.id}, (err, allMaterial) => { // Запит до БД
    if (err) {
      res.render('theme', { data: null, message: req.flash('err') })
    } else {
      res.render('theme', { data: allMaterial }) // Відправлення результатів запиту для
відображення
    }
  })
})

```

#### ЧОТИРИКУТНА ПРИЗМА



**Чотирикутна призма (куб)** - многогранник, всі грані якого - квадрати.

**Грань куба** - це частина площини, обмежена сторонами квадрата.

- куб має шість граней;
- кожна грань куба перетинається з чотирма іншими гранями під прямим кутом і паралельна шостій межі;
- межі мають однакову площу, яку можна знайти, використовуючи формули для обчислення площі квадрата.

**Ребро куба** - це відрізок, утворений перетином двох граней куба.

- куб має дванадцять ребер;
- кожен кінець ребра з'єднаний з двома сусідніми ребрами під прямим кутом;
- ребра куба мають однакову довжину.

**Вершина куба** - це найвіддаленіша від центру куба точка, яка лежить на перетину трьох граней куба.

- куб має вісім вершин;
- кожна вершина утворена тільки трьома гранями і трьома ребрами.

Рис. 4.3.2 – Результат тестування функції з відображення матеріалів

get_task	3		мокап форма
----------	---	--	-------------

```
app.get('/theme:id', (req, res) => {
  task.find({theme.id}, (err, allTask) => { // Запит до БД
    if (err) {
      res.render('theme', { data: null, message: req.flash('err') })
    } else {
      res.render('theme', { data: allTask }) // Відправлення результатів запиту для
відображення
    }
  })
})
```

#### Завдання 1

Дано призму  $ABCD A_1 B_1 C_1 D_1$ . Побудувати переріз призми площиною, що проходить через точки:  $M \in A_1 B_1 C_1 D_1$ ,  $A$  і  $D$ .

Рис. 4.3.3 – Результат тестування функції з відображення завдання

get_solution	4		мокап форма
--------------	---	--	-------------

```
app.get('/theme:id', (req, res) => {
  solution.find({task.id}, (err, allSolution) => { // Запит до БД
    if (err) {
      res.render('theme', { data: null, message: req.flash('err') })
    } else {
      res.render('theme', { data: allSolution }) // Відправлення результатів запиту для
відображення
    }
  })
})
```

▼ Розв'язання

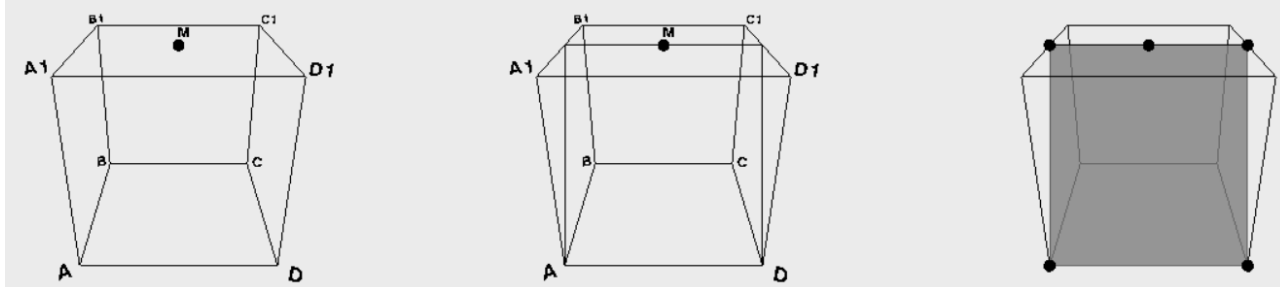


Рис. 4.3.4 – Результат тестування функції з відображення розв'язання

get_object	5		мокап форма
------------	---	--	-------------

```
app.get('/theme:id', (req, res) => {
  object.find({theme.id}, (err, allObject) => { // Запит до БД
    if (err) {
      res.render('theme', { data: null, message: req.flash('err') })
    } else {
      res.render('theme', { data: allObject }) / Відправлення результатів запиту для
відображення
    }
  })
})
```

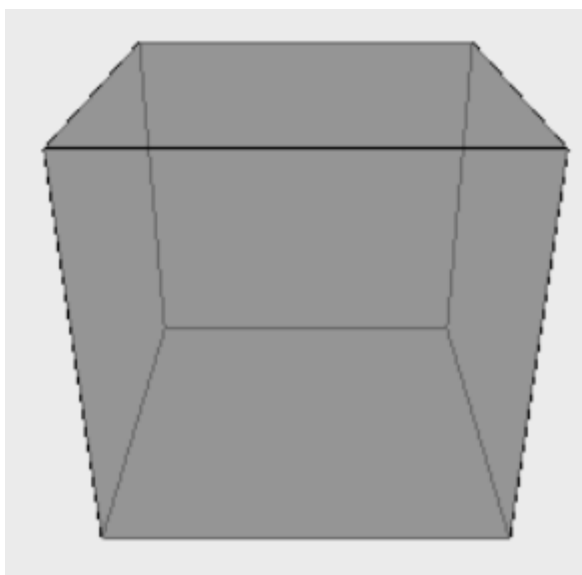


Рис. 4.3.5 – Результат тестування функції з відображення об'єкту

use_object	6		мокап форма
------------	---	--	-------------

Зміна стану об'єкта відбувається на стороні клієнта. Приклад обертання.

```
{
  document.querySelector('#cube').onclick = function(e){
    if(rotateCube==false){rotateCube = true} else {rotateCube=false}
```

```

}}

if (rotateCube==true)
{
    var time = 0.001
    matrix.makeRotationY(time * 2 * Math.PI);
    camera.position.applyMatrix4(matrix);
    camera.lookAt(0,0,0);
    quater = camera.quaternion;

}else if (rotateCube==false)
{
    camera.quaternion = quater;
}

```

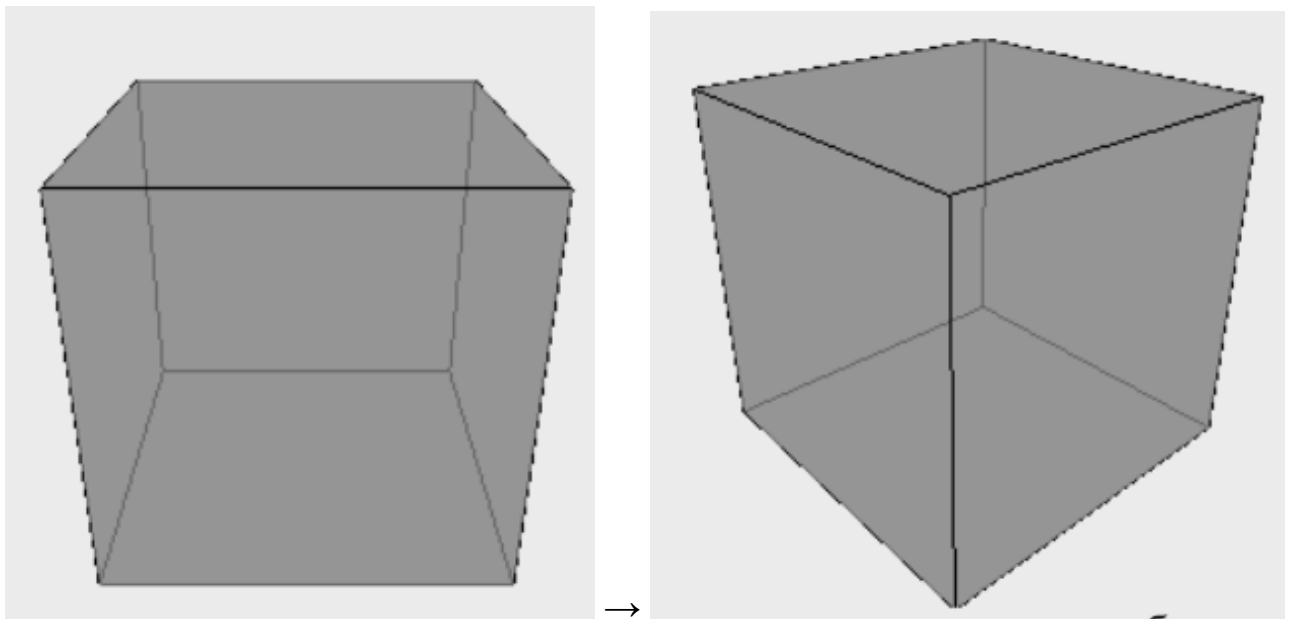


Рис. 4.3.6 – Результат тестування функції з обертання об'єкту

use_object	7		мокап форма
------------	---	--	-------------

#### Приклад побудови перерізу об'єкта

```

{
    document.querySelector('#cube').onclick = function(e){
        CubeClickCounter++;
        if(rotateCube==false){rotateCube = true} else {rotateCube=false}
    }
}

If(CubeClickCounter==2)
{
    scene.remove(Cube,EdgeCube,mesh);
    scene.add(CubeHalfCube, CubeHalfEdgeCube,CubeHalfCube2,
CubeHalfEdgeCube2);
    CubeHalfEdgeCube.position.set(-0.03, 0, -0.03);
}

```

```

CubeHalfCube.position.set(-0.03, 0, -0.03);

CubeHalfEdgeCube2.position.set(0.03, 0, 0.03);
CubeHalfCube2.position.set(0.03, 0, 0.03);

} else if(CubeClickCounter==4)
{
    scene.remove(CubeHalfCube, CubeHalfEdgeCube, CubeHalfCube2,
CubeHalfEdgeCube2);
    scene.add(EdgeCube, mesh, Cube);
    CubeClickCounter=0;
}

```

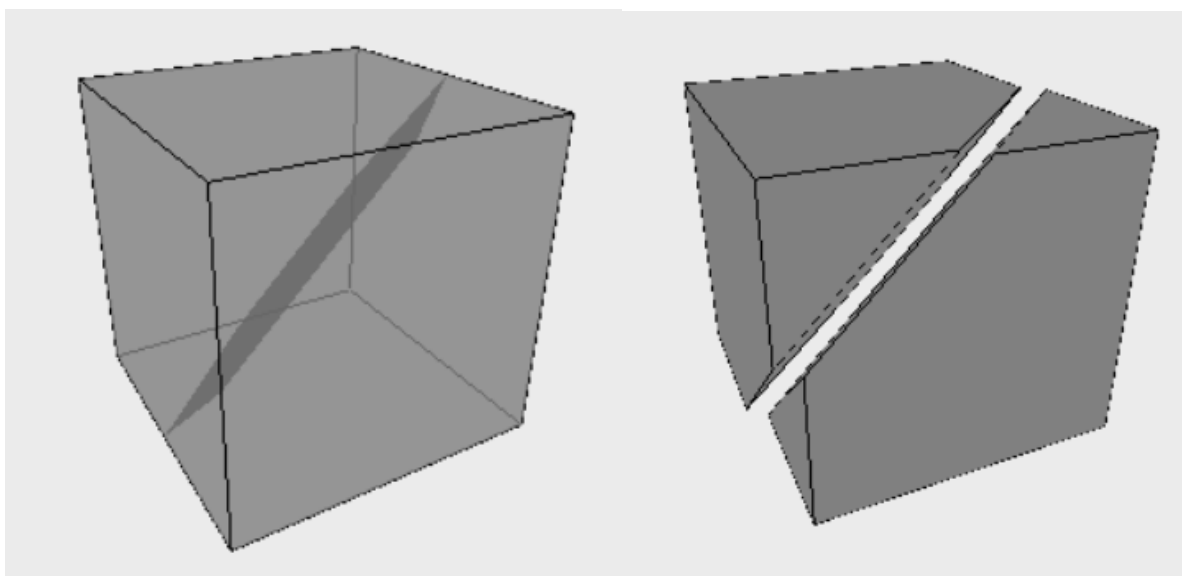


Рис. 4.3.7 – Результат тестування функції з побудови перерізу об'єкту

## 5 Розгортання та валідація програмного продукту

### 5.1 Інструкція з встановлення програмного продукту

Для встановлення програмного продукту необхідно на сервері встановити сервер node.js, БД MongoDB. Для запуску сервера необхідно в командному рядку перейти до директорії файлів node.js та виконати команду – `node index.js`, якщо запуск серверу буде успішний виведеться відповідне повідомлення. Після встановлення БД MongoDB необхідно створити БД, колекції та імпортувати файли бази(рис.5.1).

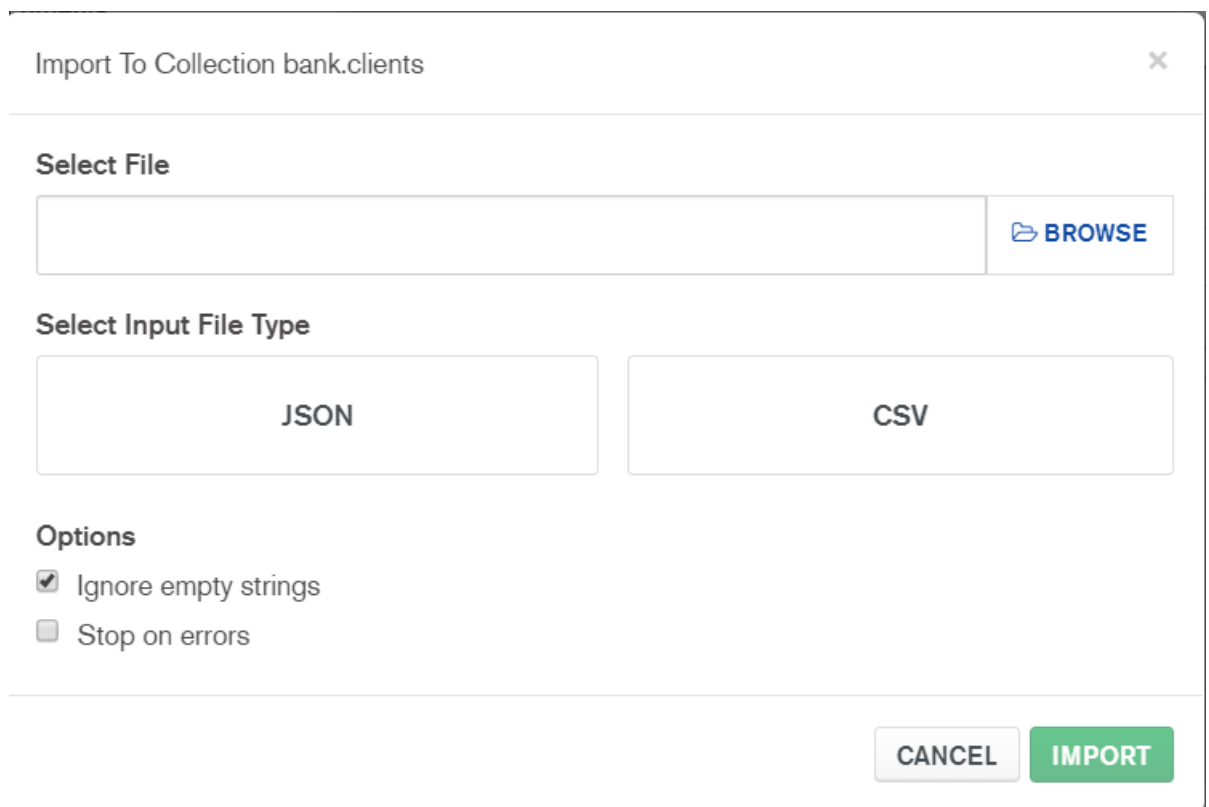


Рис. 5.1 – Імпорт файлів колекції

ПП буде розміщений за адресою – `http://localhost:3000`.

### 5.2 Інструкція з використання програмного продукту

На головній сторінці веб-додатку, у лівому верхньому куті є кнопка меню(рис.5.2.1) за допомогою неї здійснюється навігація. Після натискання на кнопку відкривається меню з вибором теми(рис.5.2.2).



Рис. 5.2.1 – Кнопка меню

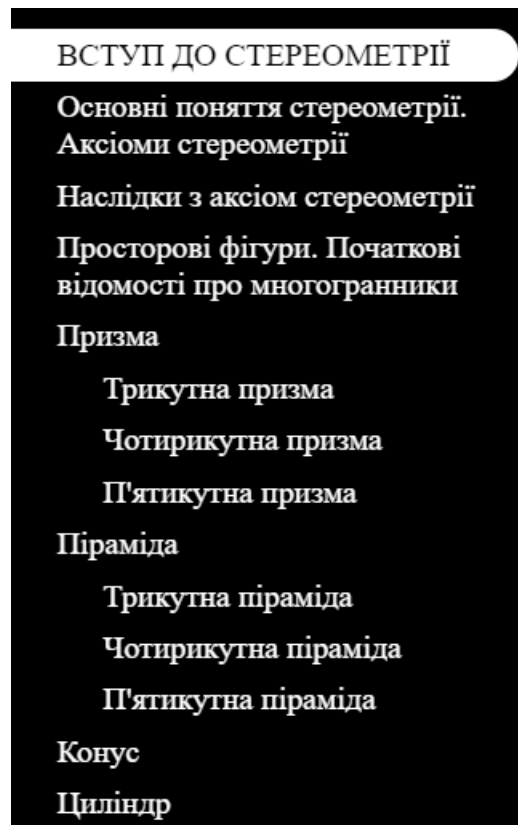


Рис. 5.2.2 – Меню

Використовуючи відповідне меню здійснюється навігація додатком. При переході на будь-який пункт меню можна бачити сполери, що ховають деяку інформацію(рис. 5.2.3), при натисканні на трикутник відбувається відкриття сполеру(рис. 5.2.4).

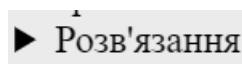


Рис. 5.2.3 – Приклад сполеру



Рис. 5.2.4 – Приклад відкритого сполеру

Також у додатку присутні деякі геометричні фігури, з ними можлива взаємодія. Для запуску обертання необхідно натиснути ПКМ 1 раз, для побудови перерізу – 2 рази.

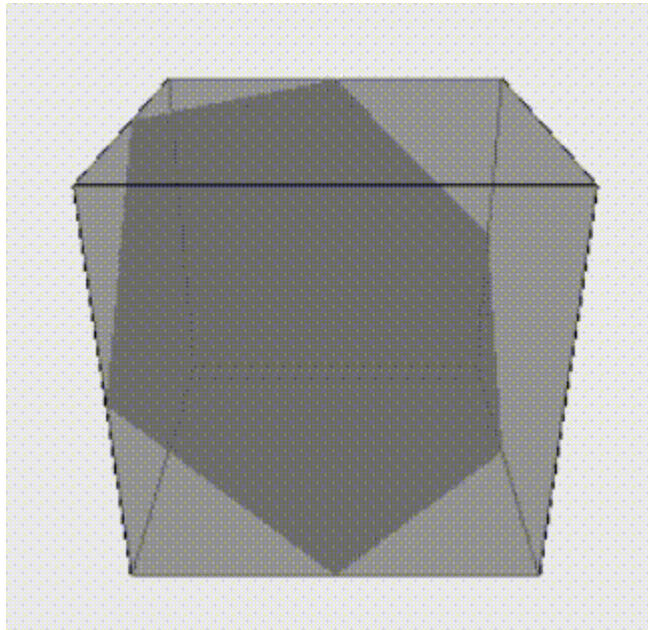


Рис. 5.2.5 – Приклад взаємодії з геометричним об'єктом

### 5.3 Результати валідації програмного продукту

Приклад аналогу ПП, що не має достатньої кількості прикладів, та взагалі не має завдань(рис.5.3.1). Схожа реалізація теми у створеному ПП(рис. 5.3.2).



**Діагональне сечення призми** — это сечение плоскостью, проходящей через два боковых ребра, не принадлежащие одной грани.

Каждое диагональное сечение содержит две диагонали призмы.

Диагональное сечение прямой призмы является **прямоугольником**.

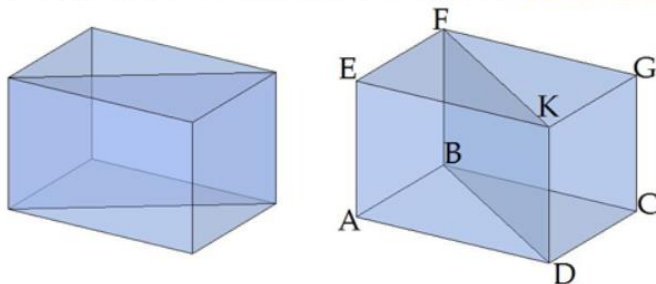
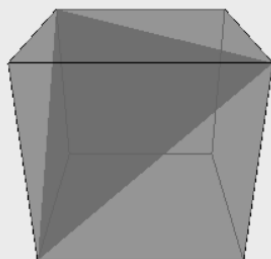
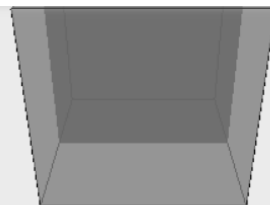


Рис. 5.3.1 – Зображення аналогу ПП



Якщо перетнути куб площиною, що проходить через центр куба і центри двох протилежних граней, то в перерізі буде квадрат, довжина сторони якого буде дорівнює довжині ребра куба. Ця площина ділить куб на два рівних прямокутних паралелепіпеда.



Якщо перетнути куб площиною, що проходить через три вершини куба, то в перерізі буде правильний трикутник. Одна з діагоналей куба перпендикулярна площині перетину і проходить через центр трикутника і ділиться площиною відповідно 2:1.

Якщо перетнути куб площиною, що проходить через центр і середини шести граней, то в перерізі буде правильний шестикутник. У куба одна з діагоналей кожної грані, що перетинаються, перпендикулярна стороні шестикутника.

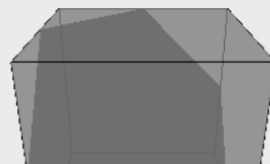


Рис. 5.3.2 – Зображення створеного ПП

Щодо кількості та якості завдань для самоконтролю, на рисунку 5.3.3 зображено приклад завдання аналогу, на рисунку 5.3.4 – створеного ПП.

Условие задания:		1 Б.
<p>Объём куба равен 125 куб. ед. Вычисли площадь полной поверхности куба.</p>		
<p><input type="radio"/> 150</p> <p><input type="radio"/> 200</p> <p><input type="radio"/> 25</p> <p><input type="radio"/> 100</p>		
<p><input type="button" value="Вход"/> или <a href="#">Регистрация</a></p>		

Рис. 5.3.3 – Приклад завдання аналогу

**Завдання 4**

Дано куб  $ABCA_1B_1C_1D_1$ . Побудувати переріз куба площиною, що проходить через точки:  $M \in AA_1DD_1$ ,  $N \in DD_1$  і  $K \in DC_1$ .

▼ Розв'язання

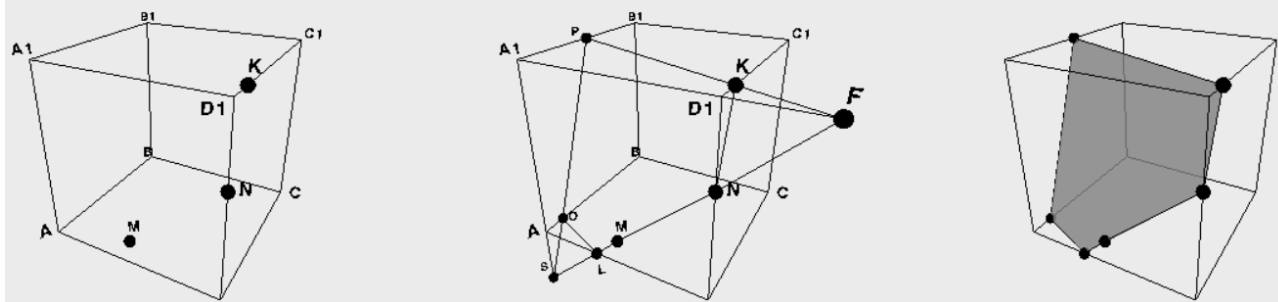


Рис. 5.3.4 – Приклад завдання та рішення у створеному ПП

Кількість елементарних дій для відкриття необхідної теми у аналогу – 5.

Кількість елементарних дій для відкриття необхідної теми у створеному ПП – 2.

## **Висновки**

В результаті створення програмного продукту була досягнута наступна мета його споживача: «визначення мети з підпункту 1.2.2.2».

Доказом цього є наступні факти:

- 1) Графічне представлення реалізоване, та зображене на рисунках 5.3.2, 5.3.4.
- 2) Кількість завдань для кожної теми не менше 3х.
- 3) Підвищення енергомичності опис у пункті 5.3.

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

- 1) Відсутність досвіду, знань у розробці серверної частини.
- 2) Відсутність досвіду, знань у роботі з графічними бібліотеками.

Через обмежений час на створення програмного продукту, залишилися нереалізованими ідеї, що можуть бути реалізовані в майбутніх курсових роботах з урахуванням тем дисциплін наступних семестрів.