

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

## **КУРСОВА РОБОТА**

з дисципліни “Технології створення програмних продуктів”  
за темою “Веб-додаток – URegion”

Виконала:  
студентка 3-го курсу  
групи AI-183  
Ткаченко К.І.  
Перевірив:  
Блажко О. А

Одеса – 2020

### **Анотація**

В курсовій роботі розглядається процес створення програмного продукту «Веб-додаток – URegion». В пояснювальній записці у розділах «Проектування» та «Конструювання» детально описано особливості конструювання структур даних реляційної бази даних PostgreSQL, а також програмних модулів в інструментальному середовищі «Visual Studio» з використанням фреймворка ASP.NET Core.

Результати роботи розміщено на github-репозиторії за адресою:  
<https://github.com/Olexsays/UPigeon>

## **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 .....	5
1.1	Визначення потреб споживача .....	5
1.1.1	Ієрархія потреб споживача .....	5
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 .....	10
1.2.3.1	Гасло програмного продукту.....	10
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 .....	12
1.3.3	Сценаріїв використання прецедентів програмного продукту .....	12
1.4.2	Функціональний аналіз існуючих програмних продуктів .....	17
1.5	Нефункціональні вимоги до програмного продукту .....	18
1.5.1	Опис зовнішніх інтерфейсів.....	18
1.5.1.1	Опис інтерфейса користувача .....	18
1.5.1.1.1	Опис INPUT-інтерфейса користувача .....	18
2	Планування процесу розробки програмного продукту.....	23

					ІС КР 122 ЕАІ185 ПЗ			
З м	А р	№.	П і д п	Д а				
		Кукурудза Д.О.			«Веб-додаток URegion»	Л і т .	Л и с т	Л и с т і в
		Блажко О.А.					3	61
						ОНПУ, каф. ІС, г. НАІ-185		
Н.								

2.1	Планування ітерацій розробки програмного продукту .....	23
2.2	Концептуальний опис архітектури програмного продукту .....	25
2.3	План розробки програмного продукту .....	26
2.3.1	Оцінка трудомісткості розробки програмного продукту.....	26
3	Проектування програмного продукту .....	31
3.1	Концептуальне та логічне проектування структур даних програмного продукту.....	31
3.1.1	Концептуальне проектування на основі UML-діаграми концептуальних класів.....	31
3.3	Проектування алгоритмів роботи методів програмних класів .....	33
3.4	Проектування тестових наборів методів програмних класів .....	35
4	Конструювання програмного продукту .....	47
4.1	Особливості конструювання структур даних .....	47
4.1.1	Особливості інсталяції та роботи з СУБД.....	47
5	Розгортання та валідація програмного продукту.....	55
5.1	Інструкція з встановлення програмного продукту.....	55
	Висновок .....	59
	ПЕРЕЛІК ПОСИЛАНЬ.....	60

## 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),
- причетність (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 соціальних потреб, так як майбутній сайт працює для підвищення ефективності пошуку знайомств з урахуванням інтересів та зручної системи по створенню та пошуку заходів по інтересам

					ІС КР 122 НАІ185 ПЗ	АР
						5
Зм.	Арк.	№ докум.	Підп.	ДА		



Рис. 1.1 – Приклад ієрархії потреби споживача

#### 1.1.2 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.

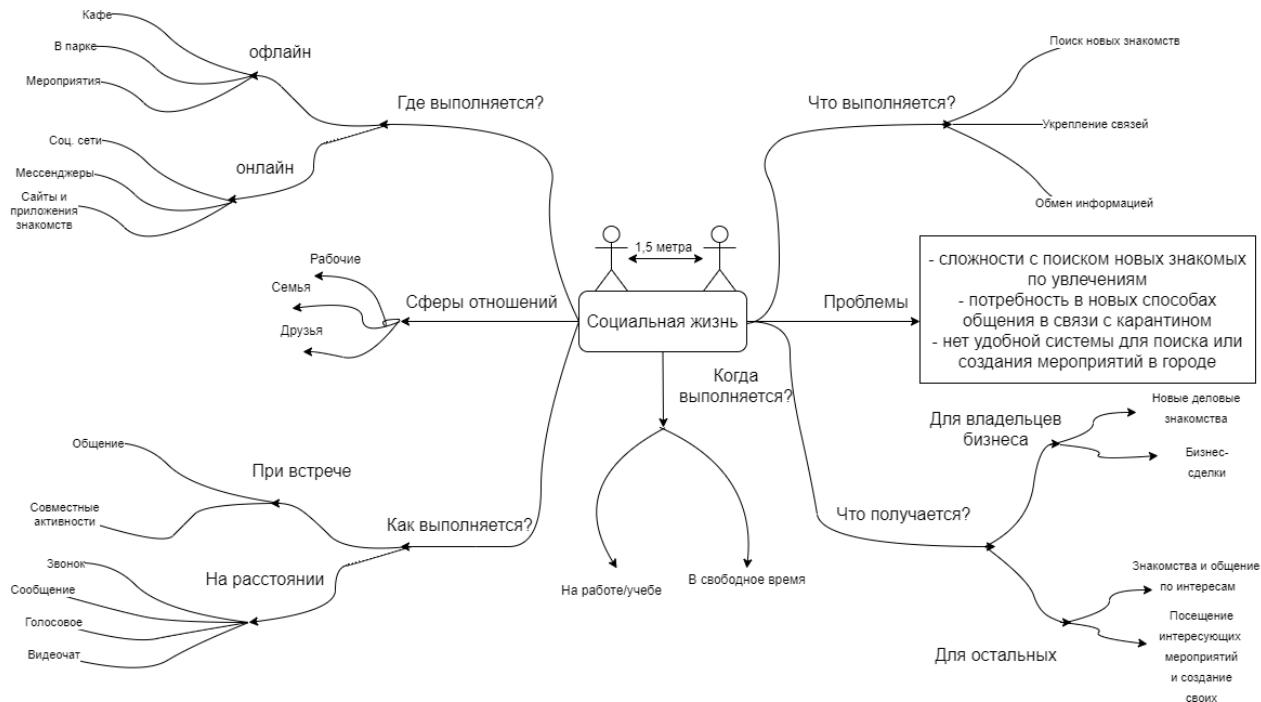


Рис. 1.2 – Приклад деталізації матеріальної потреби

## 1.2 Бізнес-вимоги до програмного продукту

### 1.2.1 Опис проблеми споживача

#### 1.2.1.1 Концептуальний опис проблеми споживача

Потреба належати до соціальної групи, причетність (спілкування, дружба, любов), у зв'язку з карантином з'являється проблема у спілкуванні і пошуку нових знайомств, отже багато людей перейдуть на різні соціальні мережі, і тут виникає інша проблема: немає зручної системи для пошуку за фільтрами, або створення заходів в місті.

#### 1.2.1.2 Метричний опис проблеми споживача

Для скорочення часу і коштів при задоволенні реальних потреб

Зм.	Арк.	№ докум.	Підп.	ДА



людині потрібна інформація, що призводить до появи інформаційної потреби. В таблиці 1 представлено опис проблеми споживача та метричні показники незадоволеності споживача.

Загальний опис проблеми	Метричні показники незадоволеності споживача
<u>Немає</u> зручних мереж для знайомств, згідно з власними інтересами, та створення власних заходів	<u>Низький рівень кількості</u> нових знайомств по інтересам та заходів у місті, та труднощі із рекламою тих заходів, котрі існують

Таблиця 1 – опис проблеми споживача та метричні показники незадоволеності споживача

Рівень доступності AL (AL – Access Level) можна визначити як  $AL = NA / N$ , де NA – кількість знайдених людей за критеріями; N – час витрачений на пошук за фільтрами.

### 1.2.2 Мета створення програмного продукту

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);

#### 1.2.2.1 Проблемний аналіз існуючих програмних продуктів

№	Назва продукту	Вартість	Ступінь готовності	Примітка
---	----------------	----------	--------------------	----------

1	Instagram	Безкоштовно	1	Немає функціоналу подій та відсутність фільтру за інтересами
2	Tinder	Безкоштовно	1	Призначений для романтичних знайомств
3	Badoo	Частково безкоштовно	1	Недостатня технічна підтримка та часті збої в алгоритмі
4	Вконтакте	Безкоштовно	1	Заблокований на території України
5	MeetUp	Частково безкоштовно	1	Незручний функціонал та проблеми з реєстрацією

Таблиця 2 – реалізація програмного аналізу існуючих програмних продуктів

#### 1.2.2.2 Мета створення програмного продукту

Метою створення програмного продукту є підвищення ефективності пошуку знайомств за допомогою додатку з урахуванням інтересів кожного користувача і наступного пошуку людей за певним фільтром, а також конструювання зручної системи по створенню та пошуку заходів по інтересам.

### 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 - «Твоє повідомлення як голубка».

#### 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 пізнаваності 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 НАІ185 ПЗ	АР
						10
Зм.	Арк.	№ докум.	Підп.	ДА		

1. Як гість, я можу зареєструватись в системі для отримання облікового запису для користування додатком.
2. Як гість, я можу увійти в систему під раніше створеної обліковим записом, для подальшої роботи.
3. Як користувач, я можу змінити дані свого облікового запису для коригування змінених або невірних даних.
4. Як користувачу, мені дозволено задати параметри власних інтересів, щоб фільтрувати записи нових знайомств.
5. Я користувач, я можу задавати критерії для пошуку інших людей, аби задовольнити потребу знайомства з іншими людьми заради певної цілі.
6. Як користувач, я можу зберігати свої фотографії в системі, щоб мати можливість показати їх іншим.
7. Як користувач, я можу створювати актуальні події, котрі в свою чергу слугуватимуть для зустрічей з різними людьми.
8. Як користувач, я маю власний рейтинг, котрий задають інші користувачі, задля оцінки тої або іншої людини, щоб скласти соціальну характеристику тої або іншої людини.
9. Як користувачу, мені дозволено дізнатись по карті актуальні події, аби вибрати найбільш привабливу як за тематикою, так і за відстанню щодо власного місцезнаходження.
10. Як рекламодавець, я маю можливість додавати в додаток рекламу, щоб мати прибуток з розповсюдження програми серед багатьох користувачів.
11. Як адміністратор, я можу управляти фотографіями та даними користувачів, так щоб контент сайту був легальним.
12. Як користувач, я можу видалити свій обліковий запис і перестати бути користувачем системи

### 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 1.3.1 представлено діаграму прецедентів ПП.

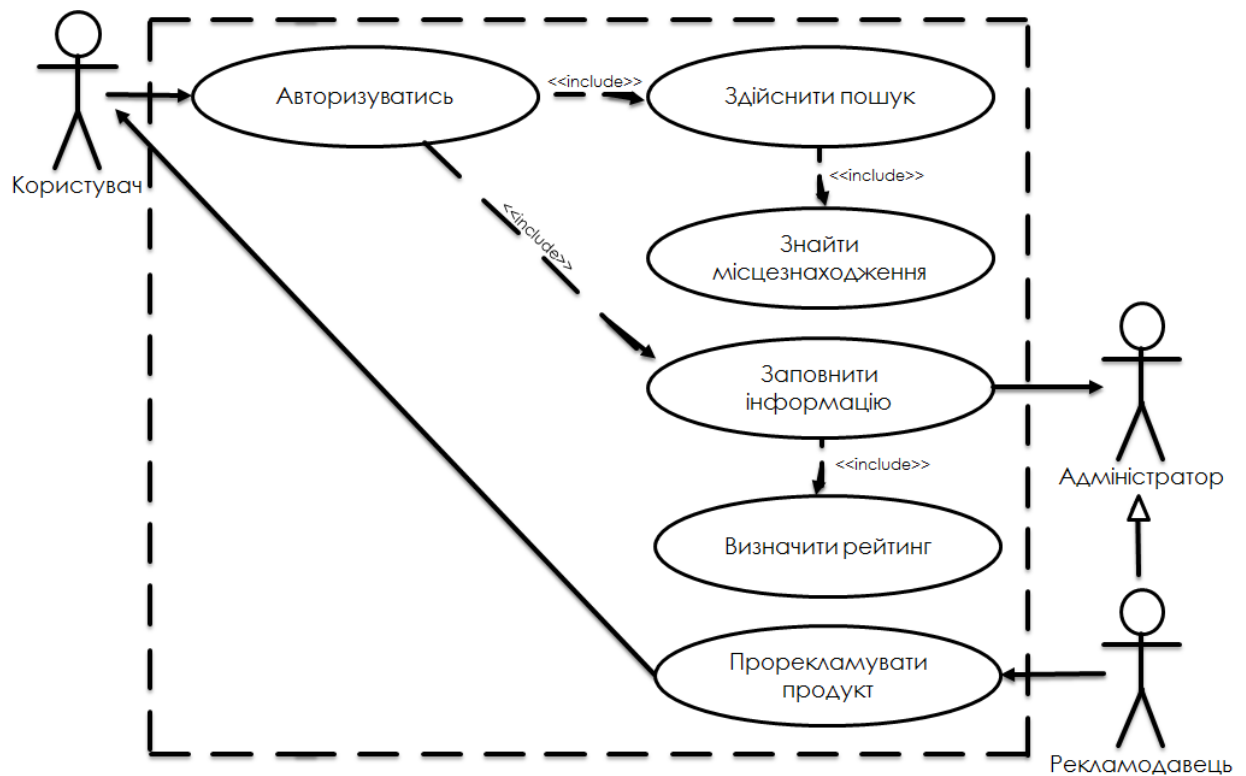


Рис 1.3.1 –Діаграма прецедентів ПП

### 1.3.3 Сценаріїв використання прецедентів програмного продукту

Опис сценаріїв:

#### 1. Авторизація:

Умова початку прецеденту: дія з боку Користувача.

Актори, що зацікавлені в виконанні даного прецеденту:  
Користувач, Адміністратор, Рекламодавець.

Актор-основа початку: Користувач.

Гарантії успіху: успішна авторизація в додаток.

Успішний сценарій:

1. ПП запитує у користувача параметри авторизації;
2. Користувач передає свої параметри;
3. ПП надає доступ.

Альтернативний сценарій:

1. ПП отримує неправильні параметри від користувача;
2. ПП виводить повідомлення про помилку.

## 2. Здійснити пошук:

- Умова початку прецеденту: здійснення авторизації.

Актори, що зацікавлені в виконанні даного прецеденту: Користувач.

Актор-основа початку: Користувач.

Гарантії успіху: здійснення пошуку по певним критеріям.

Успішний сценарій:

1. ПП запитує у користувача параметри пошуку (бажані інтереси людей для пошуку знайомств, або бажані заходи для відвідування);
2. Користувач передає бажані параметри;
3. ПП видає дані.

Альтернативний сценарій:

					ІС КР 122 НАІ185 ПЗ	АР
						13
Зм.	Арк.	№ докум.	Підп.	ДА		

1. ПП отримує неправильні параметри від користувача;

2. ПП виводить повідомлення про помилку.

**3. Знайти місцезнаходження:**

Умова початку прецеденту: здійснення пошуку.

Актори, що зацікавлені в виконанні даного прецеденту:

Користувач.

Актор-основа початку: Користувач.

Гарантії успіху: визначення правильного місцезнаходження заходу.

Успішний сценарій:

1. ПП запитує у користувача параметри пошуку;

2. Користувач передає бажані параметри для заходу та район проведення заходу;

3. ПП надає варіанти за пошуком.

Альтернативний сценарій:

1. ПП отримує неправильні параметри від користувача, або не виконується попередній прецедент;

2. ПП виводить повідомлення про помилку за неіснуючим районом пошуку або показує відсутність заходів за запитом.

**4. Заповнити інформацію:**

Умова початку прецеденту: здійснення авторизації.

Актори, що зацікавлені в виконанні даного прецеденту:

Користувач, Адміністратор.

Актор-основа початку: Користувач.

Гарантії успіху: заповнення інформації користувачем.

Успішний сценарій:

					ІС КР 122 НАІ185 ПЗ	АР
						14
Зм.	Арк.	№ докум.	Підп.	ДА		

1. ПП запитує у користувача інформацію для заповнення;
2. Користувач надає дані;
3. ПП приймає дані (Адміністратор їх перевіряє та дані не порушують умови користування додатку).

Альтернативний сценарій:

1. ПП не отримує дані від користувача;
2. ПП виводить повідомлення про помилку (Дані порушують умови користування додатку)

## 5. Визначити рейтинг

Умова початку прецеденту: прецедент «Заповнити інформацію».

Актори, що зацікавлені в виконанні даного прецеденту: Користувач.

Актор-основа початку: Користувач.

Гарантії успіху: успішне визначення рейтингу користувача ПП.

Успішний сценарій:

1. ПП визначає з заданої інформації та оцінками користувачів рейтинг заданої особи;
2. ПП показує рейтинг.

Альтернативний сценарій:

1. ПП не може визначити з заданої інформації рейтинг;
2. ПП виводить повідомлення про помилку.

## 6. Прорекламувати продукт.

Умова початку прецеденту: дія з боку Рекламодавця.

Актори, що зацікавлені в виконанні даного прецеденту:

Користувач, Рекламодавець.

Актор-основа початку: Рекламодавець.

Гарантії успіху: вивід Користувачу рекламного продукту.

					ІС КР 122 НАІ185 ПЗ	АР
						15
Зм.	Арк.	№ докум.	Підп.	ДА		



Успішний сценарій:

1. Рекламодавець передає параметри рекламного продукту(РП) ПП;
2. ПП представляє даний продукт користувачеві;

Альтернативний сценарій:

1. ПП не отримує РП від Рекламодавця;
2. ПП не виводить заданий прецедент.

Таблиця 1.2 – Класифікація функцій.

#### 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	Авторизуватись як користувач
FR 1.1	ПП запитує у користувача параметри авторизації
FR 1.2	Користувач передає свої параметри
FR 1.3	ПП надає доступ
FR 2	Здійснити пошук події
FR 2.1	ПП запитує у користувача параметри пошуку
FR 2.2	Користувач передає бажані параметри;
FR 2.3	ПП видає дані.
FR 3	Знайти місцезнаходження події
FR 3.1	ПП запитує у користувача параметри пошуку
FR 3.2	Користувач передає бажані параметри для місця

FR 3.3	ПП надає доступ
FR4	Заповнити інформацію про користувача
FR4.1	ПП запитує у користувача інформацію для заповнення
FR4.2	Користувач надає дані
FR4.3	ПП приймає дані
FR5	Визначити оцінку користувача
FR5.1	ПП визначає з заданої інформації та оцінками користувачів рейтинг заданої особи
FR5.2	ПП показує рейтинг
FR6	Прорекламувати продукт
FR6.1	Рекламодавець передає параметри рекламного продукту(РП) ПП
FR6.2	ПП представляє даний продукт користувачеві

Таблиця 3 - результат класифікації функціональних вимог

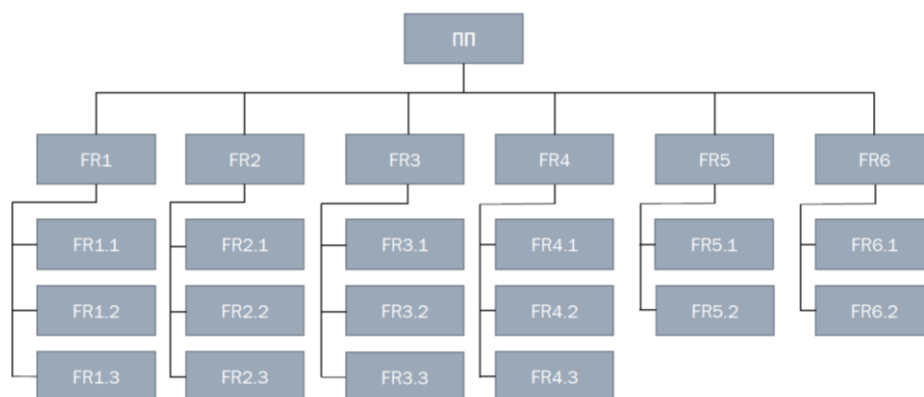


Рис 1.4 – Дерево ієрархій функцій.

#### 1.4.2 Функціональний аналіз існуючих програмних продуктів

В таблиці 4 представлено функціональний аналіз існуючих програмних продуктів.

Ідентифікатор функції	Instagram	Tinder	Badoo	Вконтакте	MeetUp
FR 1	+	+	+	+	+
FR 2	-	-	-	+	+
FR 3	+	-	-	+	-
FR 4	-	+	+	+	-
FR 5	-	+	+	-	-
FR 6	+	-	-	-	-

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

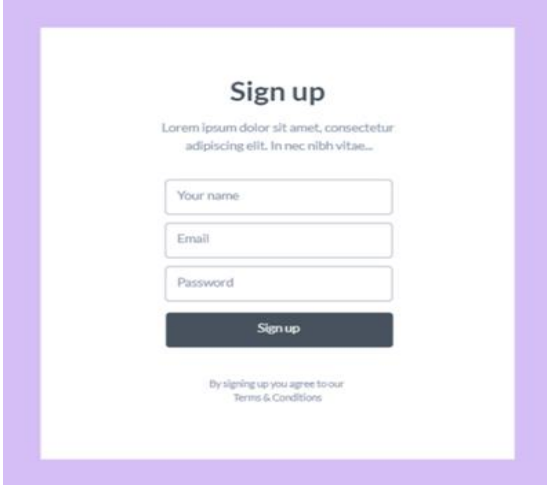
Ідентифікатор функції	Засіб INPUT-поток	Особливості використання
FR 1.1 FR 1.2	Стандартна комп'ютерна клавіатура; 2/3-кнопочний маніпулятор типу "миша«;	Використання кнопок миші чи клавіатури для введення Інформації
FR 1.3		Використання кнопок миші чи клавіатури для підтвердження
FR 2.1 FR 2.2		Використання кнопок миші чи клавіатури для підтвердження

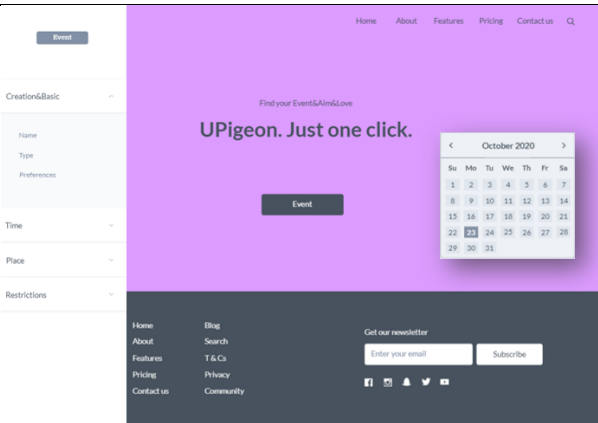
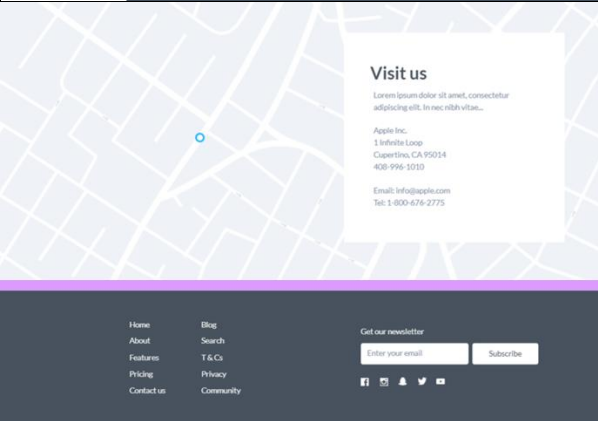
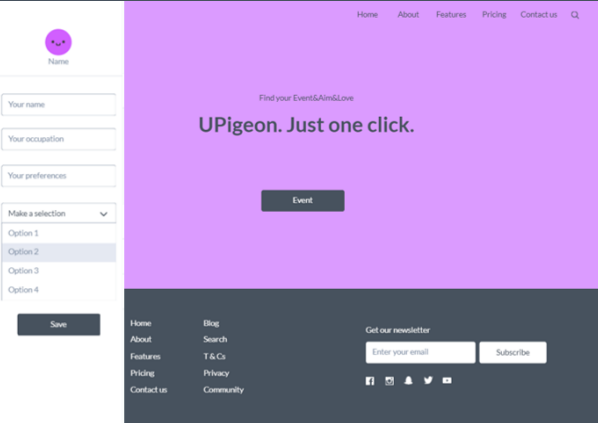
FR 2.3	сенсорний екран (Touchscreen, Touchpad, Multi-touch);	Використання кнопок миші чи клавіатури для підтвердження
FR 3.1 FR 4.2		Використання кнопок миші чи клавіатури для введення інформації
FR4.3		Використання кнопок миші чи клавіатури для підтвердження
FR5.1 FR5.2		Використання кнопок миші
FR6.1 FR6.2		Використання кнопок миші для отримання додаткової інформації

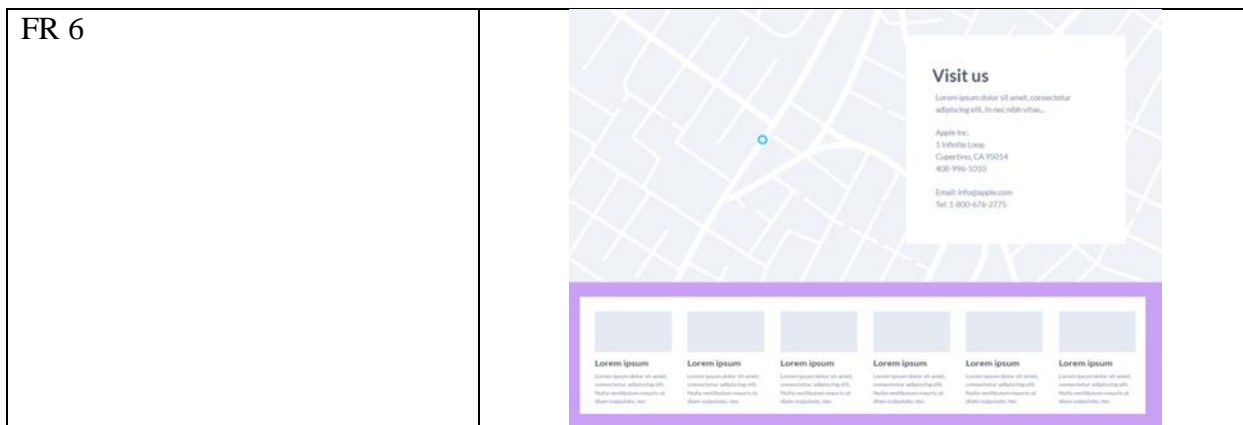
Таблиця 5 - опис INPUT- інтерфейсу користувача

#### 1.5.1.1.2 Опис OUTPUT-інтерфейсу користувача

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

Ідентифікатор функції	Засіб OUTPUT-потоків
FR 1	

FR 2	
FR 3	
FR 4	
FR 5	



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
FR1	Смартфон, планшет, Desktop-персональний комп'ютер, Notebook;
FR2	
FR3	
FR4	
FR5	
FR6	

Таблиця 7 – опис інтерфейсу із зовнішніми пристроями

### 1.5.1.3 Опис програмних інтерфейсів

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

- Windows
- Express.js
- Vue.js
- Node.js
- .NET

### 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

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.

					ІС КР 122 НАІ185 ПЗ	АР
						22
Зм.	Арк.	№ докум.	Підп.	ДА		

Ідентифікатор функції	Максимальний час реакції ПП на дії користувачів, секунди
FR 1	3
FR 2	3
FR 3	2
FR 4	2
FR 5	3
FR 6	3

Таблиця 8 – результат аналізу характеристик продуктивності

## 2 ПЛАНУВАННЯ ПРОЦЕСУ РОЗРОБКИ ПРОГРАМНОГО ПРОДУКТУ

### 2.1 Планування ітерацій 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:

– M (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;



– C (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.

Ідентифікатор функції	Назва функції	Функціональні залежності	Вплив на досягнення мети, %	Пріоритет функції
FR1	Авторизуватись як користувач	-	15	S
FR2	Здійснити пошук події	FR1	30	M
FR3	Знайти місцезнаходження події	FR2	5	W
FR4	Заповнити інформацію про користувача	FR1	10	S
FR5	Визначити оцінку користувача	FR4	5	W
FR6	Прорекламувати продукт	FR1	5	C
FR7	Здійснити пошук нових знайомств	FR1	30	M

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 2.2.1 представлено концептуальний опис архітектури програмного продукту.

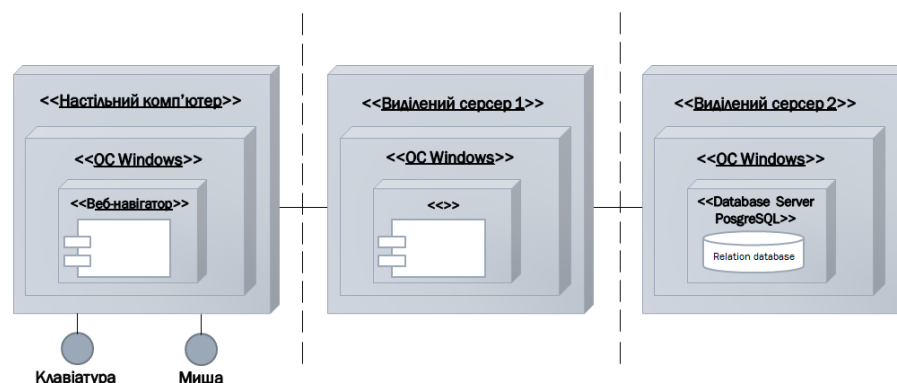


Рис 2.2.1 – концептуальний опис архітектури програмного продукту.

## 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 мережових 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
Адміністратор	Середній	2
Рекламодавець	Простий	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 прецеденту	Тип прецеденту	Кількість кроків сценарію	Ваговий коефіцієнт
------------------	----------------	---------------------------	--------------------

FR1	Середній	4-7	10
FR2	Складний	> 7	15
FR3	Простий	<= 3	5
FR4	Середній	4-7	10
FR5	Простий	<= 3	5
FR6	Простий	<= 3	5
FR7	Складний	> 7	15

Таблиця 11 – вагові коефіцієнти прецедентів

**Визначення UUCP :**

$$A = 3 + 2 + 1 = 6$$

$$UC = 10 + 15 + 5 + 10 + 5 + 5 + 15 = 65$$

$$UUCP = A + UC = 71$$

#### 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 ST<sub>i</sub> в діапазоні від 0 до 5:

- 0 означає відсутність значимості показника для даного проекту,
- 5 - високу значимість.

Значення TCF обчислюється за формулою

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

Показник	Опис показника	Вага
----------	----------------	------

T1	Розподільна система	1
T2	Висока продуктивність	2
T3	Робота користувачів в режимі онлайн	3
T4	Складність обробки даних	1
T5	Повторне використання коду	0,5
T6	Простота установки	2
T7	Простота використання	2
T8	Переносимість	1
T9	Простота внесення змін	1,5
T10	Паралелізм	0
T11	Спеціальні потреби к безпеки	1
T12	Доступ к системи зі сторони зовнішніх користувачів	3
T13	Спеціальні потреби до навчання користувачів	2

Таблиця 12 - показники технічної складності проекту TCF

**Визначення TCF :**

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

$$TCF = 0,6 + (0,01 * (20)) = 0,8$$

Показник	Опис показника	Вага
F1	Знакомство з технологією	1,5

F2	Досвід розробки додатків	1,5
F3	Досвід використання об'єктно-орієнтованого підходу	1
F4	Наявність ведучого аналітика	4
F5	Мотивація	4
F6	Стабільність потреб	2
F7	Часткова занятість	1
F8	Складні мови програмування	1

Таблиця 13 -показники рівня кваліфікації розробників

**Визначення EF :**

$$TCF = 1,4 + (-0,03 * (SF_i * Вага_i)) = 0,92$$

$$UUCP = 71$$

$$TCF = 0,8$$

$$TCF = 0,92$$

$$UCP = UUCP * TCF * EF = 52,256$$

#### 2.3.1 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 2.2

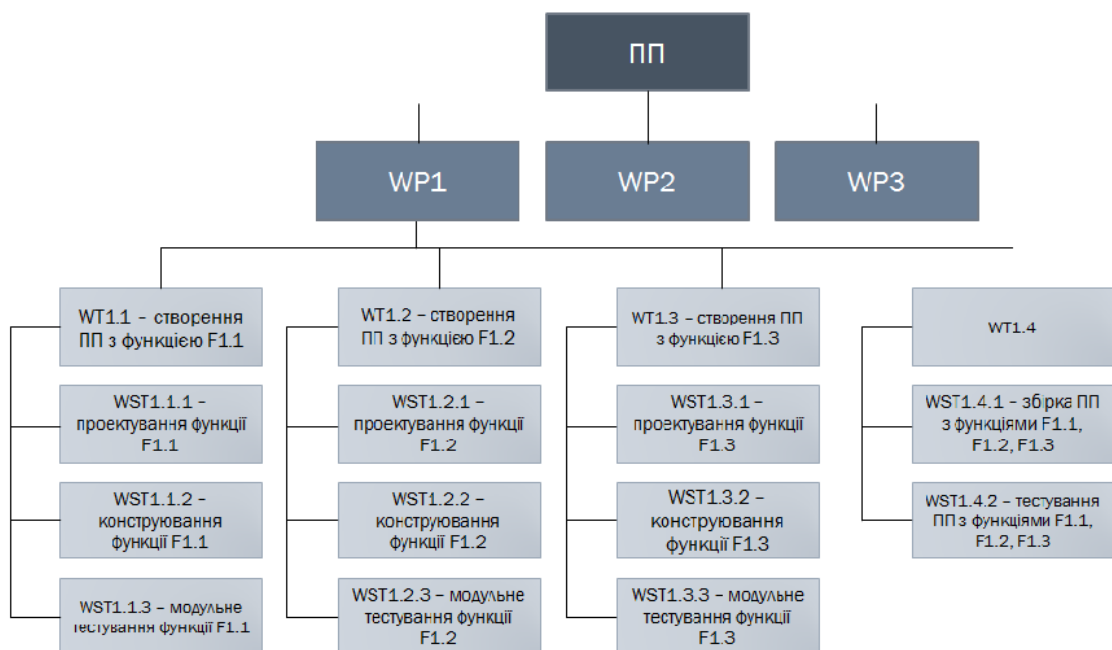


Рис. 2.2 – Приклад WBS

### 2.3.2 Графік робіт з розробки програмного продукту на рисунку 2.3:

WST	Дата початку	Дні	Дата завершення	Виконавець
1.1.1.	06.10.2020	2	07.10.2020	Кукурудза Д.О.
1.1.2.	07.10.2020	1	07.10.2020	Кукурудза Д.О.
1.1.3.	08.10.2020	2	09.10.2020	Кукурудза Д.О.
1.2.1.	10.10.2020	3	12.10.2020	Корольов О.К.
1.2.2.	11.10.2020	2	13.10.2020	Корольов О.К.
1.2.3.	12.10.2020	1	12.10.2020	Корольов О.К.
1.3.1.	13.10.2020	1	13.10.2020	Кукурудза Д.О.
1.3.2.	14.10.2020	2	15.10.2020	Кукурудза Д.О.
1.3.3.	14.10.2020	2	15.10.2020	Кукурудза Д.О.
1.4.1.	15.10.2020	1	15.10.2020	Кожушков Б. В.
1.4.2.	15.10.2020	1	15.10.2020	Кожушков Б. В.
2.1.1.	08.10.2020	2	09.10.2020	Корольов О.К.
2.1.2.	09.10.2020	1	09.10.2020	Корольов О.К.
2.1.3.	10.10.2020	1	10.10.2020	Корольов О.К.
2.2.1.	08.10.2020	2	09.10.2020	Корольов О.К.
2.2.2.	10.10.2020	2	11.10.2020	Кукурудза Д.О.
2.2.3.	11.10.2020	2	13.10.2020	Кукурудза Д.О.
2.3.1.	12.10.2020	1	12.10.2020	Кожушков Б. В.
2.3.2.	13.10.2020	1	13.10.2020	Кожушков Б. В.
2.3.3.	14.10.2020	2	15.10.2020	Кожушков Б. В.
2.4.1.	14.10.2020	2	15.10.2020	Кожушков Б. В.
2.4.2.	15.10.2020	2	16.10.2020	Кожушков Б. В.

Рис. 2.3 - Таблиця з графіком робіт

Для кожної підзадачі визначається виконавець, що фіксується у вигляді таблиці, приклад якої представлено в таблиці 13. Таблиця 14 - 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 2.4

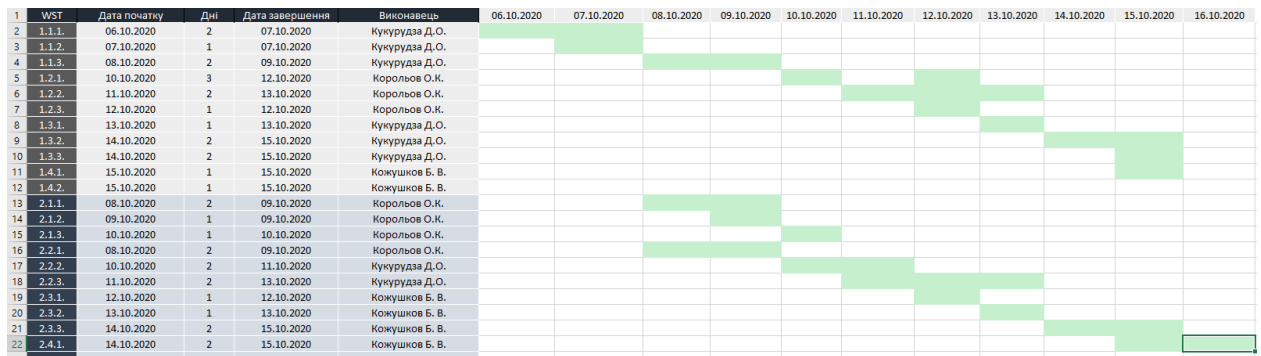


Рис. 2.4 – Приклад таблиці графіка робіт  
з ПРОЕКТУВАННЯ ПРОГРАМНОГО ПРОДУКТУ

3.1 Концептуальне та логічне проектування структур даних програмного продукту.

3.1.1 Концептуальне проектування на основі UML-діаграми концептуальних класів.

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, Інформація, Події, Реклама, Адміністратор, Знайомства, Місце.



2. Визначено імена атрибутів класів: Логін, Пароль, Опис, Розташування.
3. Визначено зв'язку між класами: 1 до N, 1 до 1.
4. Створено UML-діаграму класів, яку представлено на рисунку 3.1 представлено UML-діаграму класів.

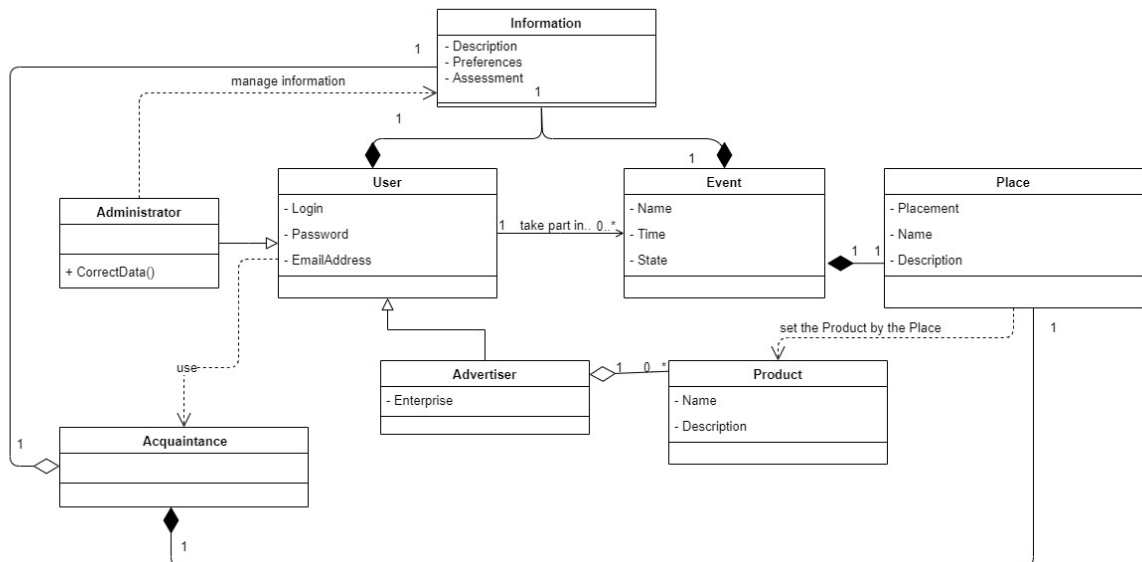


Рис. 3.1.1 - UML-діаграма класів

### 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 3.2 представлено дану діаграму.

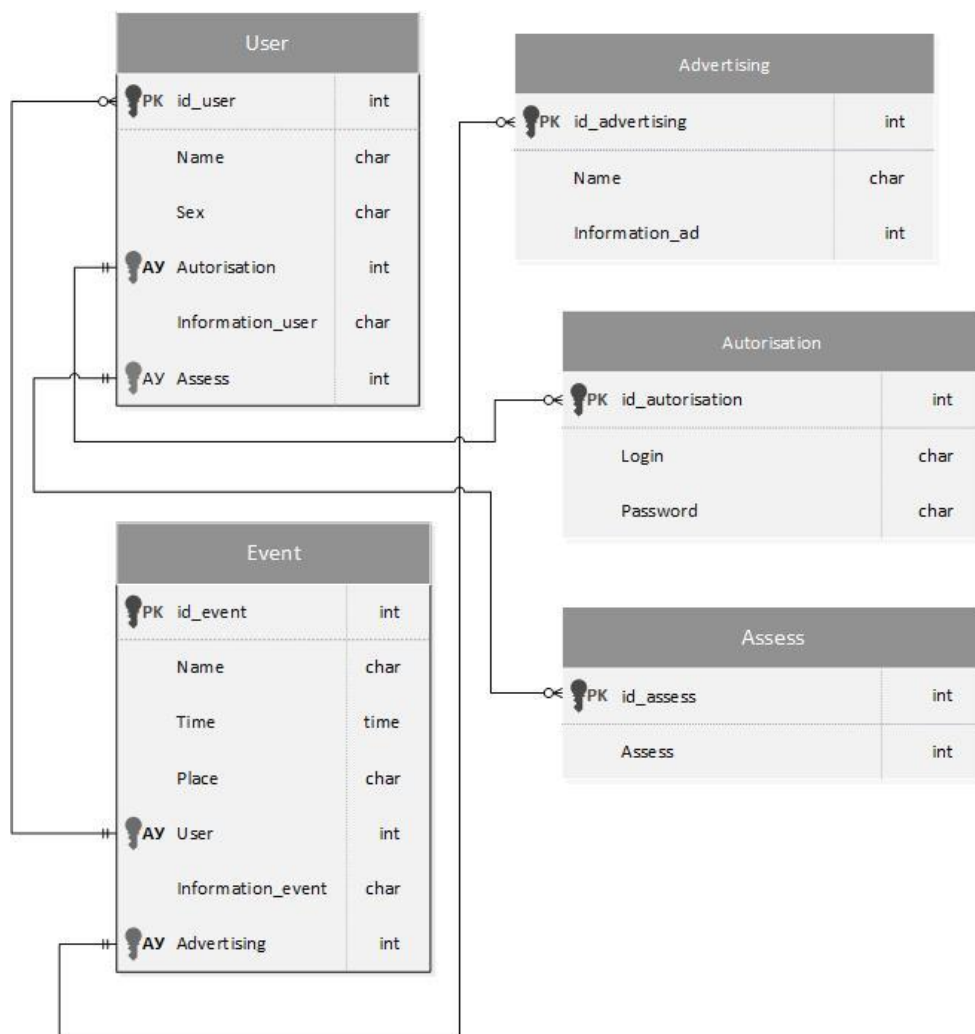


Рис. 3.1.2 – діаграма логічного проектування даних

### 3.3 Проектування алгоритмів роботи методів програмних класів

### 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 3.3 представлено проектування UML-діаграми програмних класів.

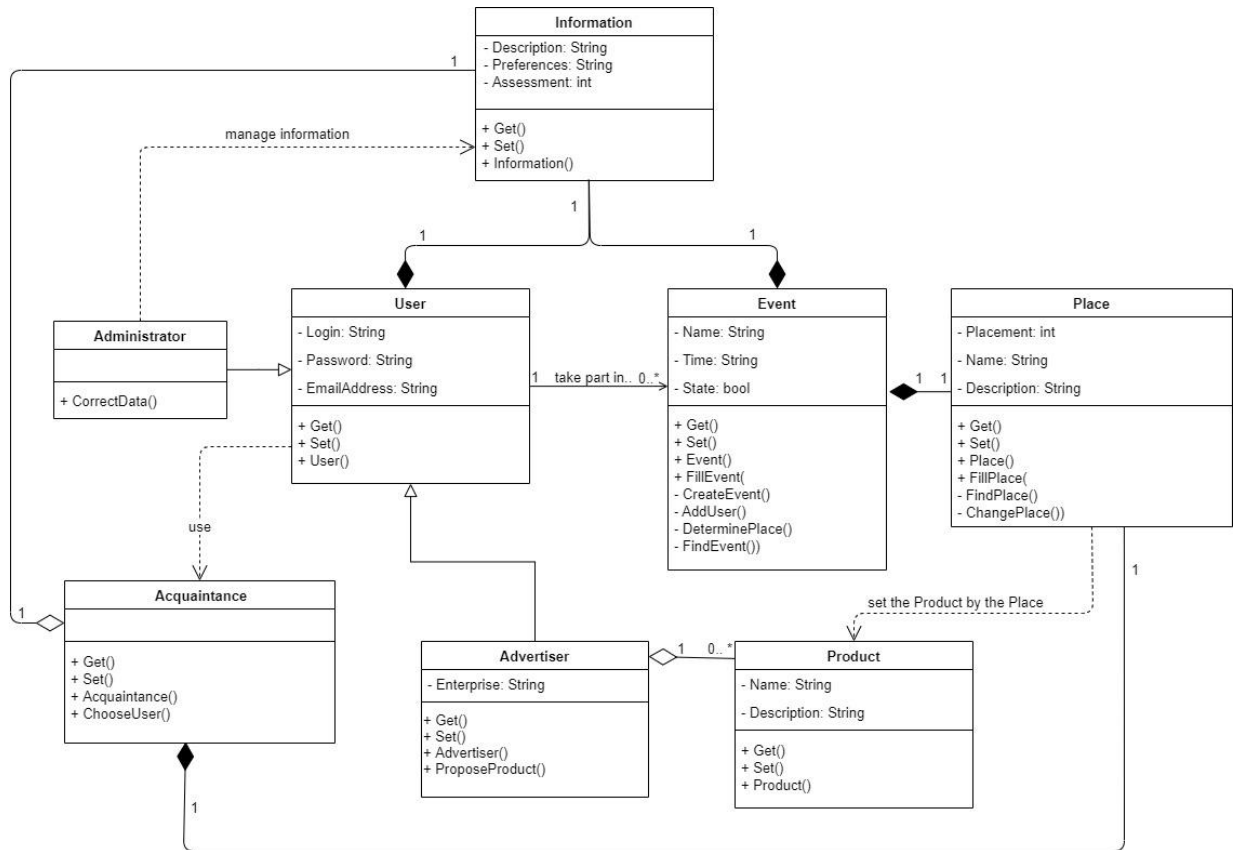


Рис 3.2.1 - проектування UML-діаграми програмних класів

### 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:

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

Далі представлений результат роботи проектування алгоритмів роботи методів програмних класів.

На рисунку 3.3.1 представлено проектування UML-діаграми активності методу AddUser, що є представленим у вигляді:

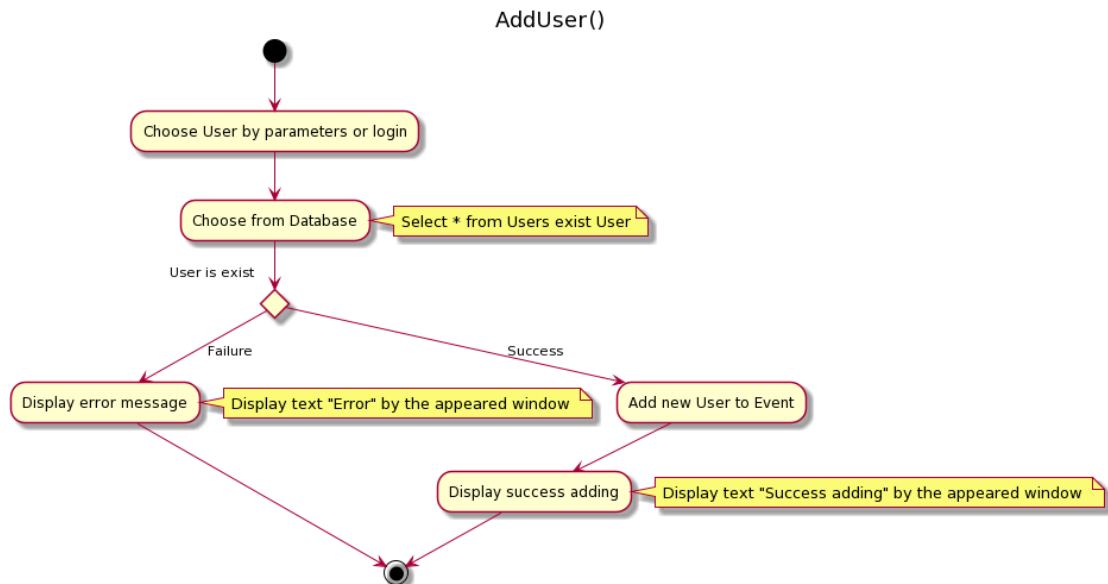


Рис 3.3.1 - проектування UML-діаграми активності методу AddUser

### 3.4 Проектування тестових наборів методів програмних класів

#### 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 вибирати найбільш підходящі підмножини, підмножини з найвищою імовірністю виявлення помилок.

Результат представлений в таблиці 15.

Назва функції	№ тесту	Опис значень вхідних даних	Опис очікуваних значень результату
Registration()	1	{ }	{ "status": 404, "message": "Error! Parameters cannot be empty" }
	2	{ "Login": "Dove", "Name": "First name": "Date of Birth": "Password": "12321qwerty" "Email": }	{ "status": 404, "message": "Error! Parameters cannot be empty" }
	3	{ "Login+": "Dove%", "Name": "First name": "Date of Birth": "Password": "12321qwerty" "Email": }	{ "status": 404, "message": "Nickname, last name, or first name contains invalid characters ('<', '>', '@', '+', '-', '/', '\', '%', '&', '?', '!', ';', ',')" }
	4	{ "Email": "qwerty.com" }	{ "status": 404, "message": "Error! Mail entered incorrectly" }
	5	{ "Login": "Qwerty" }	{ "status": 404, "message": "A user with this nickname already exists" }

			}
	6	{ "Email": "qwerty.com" }	{ "status": 404, "message": "A user with this email address already exists" }
	7	{ "Login": "qw" }	{ "status": 404, "message": "Errors! Parameters is less than 3 characters" }
	8	{ "Password": "12321qwerty" }	{ "status": 404, "message": "Error! Password must consist of 8 or more characters of the Latin alphabet, symbol"_"", contain upper and lower case letters, numbers " }
	9	{ "Login": "Dove", "Name": "qwertyqwertyqweqrtyqwertyq wertyqwerty" "First name": "Date of Birth": "Password": "12321Q_werty" "Email": }	{ "status": 404, message": "Nickname or surname, name and email address exceeds character limit (> 35)" }
	10	{ "Date of Birth": "28.08.2012" }	{ "status": 404, "message": "User's age cannot be below 12 and above 99" }
	11	{ "Login": "Dove", "Name": "Nick", "First name": "Durov", "Age": "41" "Password": "12321Q_werty" }	{ "status": 202, "message": "Successful! User add" }

		“Email”: <a href="mailto:qwerty@gmail.com">qwerty@gmail.com</a> , }	
Authorization()	1	{ “Login”: “Dove”, “Password”: “12321Q_werty” }	{ “status”: 202, “message”: “Successful login” }
	2	{ “Login”: “Dove”, “Password”: “12321Q_werty” }	{ “status”: 404, “message”: “Error! There are no users with the specified nickname on the site” }
	3	{ “Login”: “Dove”, “Password”: “12321Q_werty” }	“status”: 404, “message”: “Error! The password or login is incorrect.” function (ifPaswordOrLoginF orgotten) }
ifPaswordOrLoginForgotten	1	{ “Email”: “qwerty.com” }	{ “status”: 404, “message”: “Error! There are no users with the e-mail entered on the site” }
	2	{ “Email”: “qwerty.com” }	{ “status”: 404, “message”: “Check youre e-mail, an e- mail with the ability to change the login or password was sent to the e-mail specified by the user” }
fillUserInforamtion	1	{} }	{ “status”: 404, “message”: “Error! Parameters cannot be empty” }

	2	{ //максимум 10 атрибутів "Attribute 1": спорт, "Attribute 2": программирование, "Attribute n":... "Description": Runner / Dancer / Programmer }	{ "status": 202, "message": "Successful!" }
	3	{ //максимум 10 атрибутів "Attribute 1": спорт, "Attribute 2": программирование, "Attribute n":... "Description": "Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum." }	{ "status": 404, "message": "Error! "Description can contain up to 300 characters" }
changePassword	1	{ "Old password": "12321Q_werty" "Password": "12321Q-WERTY" "Retrypassword": "12321Q- WERTY" }	{ "status": 404, "message": "Error! Password must consist of 8 or more characters of the Latin alphabet, symbol" _", contain upper and lower case letters, numbers " }
	2	{ "Old password": "12321Q_werty"	{ "status": 404, "message": "Error!"



		"Password": "12321Q_WERTY" "Retrypassword": "12321QWERTY" }	the repeated password does not match the new password"
	3	{ "Old password": "2321Q_werty" "Password": "12321Q-WERTY" "Retrypassword": "12321Q-WERTY" }	{ "status": 404, "message": "Error! Invalid old password" }
changeLogin()	1	{ "new_login": "Turtle1" "Password": "12321Q_werty" }	{ "status": 202, "message": "Successful!" }
	2	{ "new_login": "Turtle" "Password": "2321Q_werty" }	{ "status": 404, "message": "Error! Invalid password" }
	3	{ "new_login": "Turtle" "Password": "2321Q_werty" }	{ "status": 404, "message": "Error! There are no users with the specified nickname on the site" }
	4	{ "new_login": "%Turtle%" "Password": "12321Q_werty" }	{ "status": 404, "message": "Nickname contains invalid characters (<, >, '@', '+', '-', '/', '\\', '%', '&', '?', '!', ';', ':')" }
leaveFromAccount	1	Користувач натискає кнопку виходу із акаунту	Правильний клас: Користувач сайту стає гостем, та з його комп'ютеру не можна керувати його акаунтом без авторизації.

	2	Користувач натискає кнопку виходу із акаунту.	Неправильний клас: Дані акаунта залишились.
deleteAccount	1	Користувач натискає кнопку видалення акаунту	/ Правильний клас: Акаунт з усіма даними користувача видаляється.
	2	Користувач натискає кнопку видалення акаунту	Неправильний клас: Дані про акаунт частково залишаються та інші користувачі можуть із ним контактувати.
FindAcquaintanceByKeyP references	1	//Користувач заповнює форму пошуку нових знайомств за інтересами, де указує бажаний вік, місце знаходження, та захоплення іншої людини по ключовим словам  "Date of Birth": 11.11.2001 "Attribute 1": спорт, "Attribute 2": программирование "Location": Odessa, primorsky district	{ "status": 202, "message": "Successful!" } На сайті користувачу показуються всі відповідні його запросу інші користувачі, яким він може надати запрос на переписку.
	2	//Користувач заповнює форму пошуку нових знайомств за інтересами, де указує бажаний вік, місце знаходження, та захоплення іншої людини по ключовим словам  "Date of Birth": 11.11.2001 "Attribute 1": спирт, "Attribute 2": бер "Location": Odessa, primorsky district	{ "status": 404, "message": "Error! there are no users for the specified criteria, or the criteria are specified incorrectly" }
sendMessageToUser	1	Користувач пише бажаний текст, або обирає бажаний файл та натискає на кнопку відправки.	{ "status": 202, "message": "Successful!" }

	2	Користувач пише бажаний текст, або обирає бажаний файл та натискає на кнопку відправки.	{ "status": 404, "message": "Error! Message is too large (more than 1000 characters)." }
	3	Користувач пише бажаний текст, або обирає бажаний файл та натискає на кнопку відправки.	{ "status": 404, "message": "Error! The site does not support sending files of a user-specified format } }
	4	Користувач пише бажаний текст, або обирає бажаний файл та натискає на кнопку відправки	Користувач що відправляє повідомлення заблокований адресатом. { "status": 404, "message": "Error! User is blocked by the recipient" }
FindEventByLocation	1	Користувач вводить у відповідній формі город, його вулицю або конкретну адресу для пошуку івентів. { "city": Odessa "Region": Odessa "District": Primorski "Street": Preobrajenska 21 }	Користувачу виводиться список усіх івентів, що підходять. Під критерії пошуку { "status": 202, "message": "Successful!" }
	2	Користувач вводить у відповідній формі город, його вулицю або конкретну адресу для пошуку івентів. { "city": Odessa "Region": Odessa "District": Primorski "Street": Preobrajenska 503 }	{ "status": 404, "message": "Error! The city, street, distinct or specific address entered by the user does not exist" }

		}	
	3	Користувач вводить у відповідній формі город, його вулицю або конкретну адресу для пошуку івентів. { "city": Odessa "Region": Odessa "District": Primorski "Street": Preobrajenska 503 }	{ "status": 404, "message": "Error! There are no relevant events on the site on request." }
FindEventByKeyAttributes	1	Користувач вводить відповідні ключові атрибути для пошуку подій за ними. { "Attribute 1": "спорт", "Attribute 2": "кіно", "Attribute 3": "ставки", "Attribute 4": "бар" }	Правильний клас: користувачу виводиться список усіх івентів, що підходять. { "status": 202, "message": "Successful!" }
	2	Користувач вводить відповідні ключові атрибути для пошуку подій за ними. { "Attribute 1": "спорт", "Attribute 2": "кіно", "Attribute 3": "ставки", "Attribute 4": "бар" "Attribute 5": "тиша" }	Неправильний клас: За запитом не існує відповідних подій на сайті { "status": 404, "message": "Error! There are no relevant events on the site on request" }
createEvent	1	Користувач обирає тему події із списку ключових атрибутів, задає їй назву, також задає віковий діапазон та опис події { "Event name": "Shh, don't disturb", "Event description": "Opening hookah for sociopaths", "Age restrictions": "18+", "Attribute 1": "тихо", "Attribute 2": "затишно", "Attribute 3": "hookah" "Attribute 3": "приватність"	Правильний клас: відповідна подія створюється та доступна для перевірки адміністратором { "status": 202, "message": "Successful! The event is under consideration by the administration" }

		“Location”:”Сегетська 2”, }	
	2	Користувач обирає тему події із списку ключових атрибутів, задає їй назву, також задає віковий діапазон та опис події { “Event name”: ”Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo”, “Event description”: “Opening hookah for sociopaths”, “Age restrictions”:”18+”, "Attribute 1": “тихо”, "Attribute 2": “затишно”, "Attribute 3": “hookah” "Attribute 3": “приватність” “Location”:”Сегетська 2”, }	Неправильний клас: тема події перевищує 100 символів, або опис перевищує 1000 символів. Виводиться повідомлення про помилку.  { “status”: 404, “message”: “Error! Theme of the event exceeds 100 characters, or the description exceeds 1000 characters” }
	3	{ “Event name”: ”Shh, don’t disturb”, “Event description”: “Opening hookah for sociopaths”, “Age restrictions”:”18+”, "Attribute 1": “тихо”, "Attribute 2": “затишно”, "Attribute 3”: “кальянчикус” "Attribute 3”: “приватність” “Location”:”Сегетська 2”, }	Неправильний клас: відповідних ключових атрибутів не існує.  { “status”: 404, “message”: “Error! Ehe corresponding key attributes do not exist or are entered incorrectly” }
	4	{ “Event name”: ”Shh, don’t disturb”, “Event description”: “”, “Age restrictions”: "Attribute 1”: “тихо”, "Attribute 2”: “затишно”, "Attribute 3”: “hookah” "Attribute 3”: “приватність” “Location”:”Сегетська 2”, }	{ “status”: 404, “message”: “Error! All fields must be filled” }

deleteEvent	1	Користувач, що має доступ до управління подією (або адміністратор) натискає кнопку видалення події.	Правильний клас: подія видалена, та не може бути знайдена за загальним пошуком.  { “status”: 202, “message”: “Successful!” }
	2	Користувач, що має доступ до управління подією (або адміністратор) натискає кнопку видалення події.	Дані про подію частково залишаються та інші користувачі можуть її знайти.  { “status”: 404, “message”: “Error! Try again later” }
addManagerToEvent	1	Користувач, що створив подію натискає на кнопку додавання користувача до управління подією, та обирає користувача із списку друзів.	Правильний клас: Подією можуть керувати усі, кого додав творець події.
	2	Користувач, що створив подію натискає на кнопку додавання користувача до управління подією, та обирає користувача із списку друзів.	Неправильний клас: Аккаунт, що пробує додати користувач був видалений.  { “status”: 404, “message”: “Error! This account has been deleted” }
approveEvent	1	Адміністратор натискає на кнопку підтвердження події	Правильний клас: тепер подію можна знайти в загальному пошуку  { “status”: 202, “message”: “Successful! Event confirmed”

			}
	1	Адміністратор натискає на кнопку підтвердження події	Неправильний клас: у параметрах події були недопустимі значення  { "status": 404, "message": "Error! Unacceptable value" }
deleteUser	1	Адміністратор натискає на кнопку видалення користувача	Правильний клас: Аккаунт з усіма даними користувача видалюється
	2	Адміністратор натискає на кнопку видалення користувача	Неправильний клас: Дані про аккаунт частково залишаються та інші користувачі можуть із ним контактувати.
addRating	1	Користувач нажимає на кнопку лайка у події	Правильний клас: Рейтинг користувача, що створив подію збільшується на один пункт
	2	Користувач нажимає на кнопку лайка у події	Неправильний клас: Рейтинг користувача є максимальним (100). Нічого не змінюється.
declineRating	1	Користувач натискає на кнопку дізлайку у події	Правильний клас: Рейтинг користувача, що створив подію зменшується на один пункт
	2	Користувач натискає на кнопку дізлайку у події	Неправильний клас: Рейтинг користувача є мінімальним (-100). Нічого не змінюється.

Таблиця 15 – 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 - PostgreSQL — широко розповсюджена система керування базами даних з відкритим сирцевим кодом. Прототип був розроблений в Каліфорнійському університеті Берклі в 1987 році під назвою POSTGRES, після чого активно розвивався і доповнювався. В червні 1990 року з'явилась друга версія із переробленою системою правил маніпулювання та роботи з таблицями, у 1991 році — третя версія, із доданою підтримкою одночасної роботи кількох менеджерів збереження, покращеним механізмом запитів і доповненою системою внутрішніх правил.

Переваги PostgreSQL:

1. підтримка БД необмеженого розміру;
2. потужні і надійні механізми транзакцій і реплікації;
3. розширювана система вбудованих мов програмування і підтримка завантаження Сумісних модулів;
4. спадкування;
5. легка розширюваність.

#### 4.1.2 Особливості створення структур даних

Нижче представлений код роботи структури даних PostgreSQL при обробці помилок:

```
do language plpgsql $$
declare
  v_state TEXT;
  v_msg TEXT;
  v_detail TEXT;
  v_hint TEXT;
  v_context TEXT;
begin
```

					ІС КР 122 НАІ185 ПЗ	АР
						47
Зм.	Арк.	№ докум.	Підп.	ДА		



```

create table messages (
  id_message SERIAL NOT NULL PRIMARY KEY,
  from_user INTEGER NOT NULL REFERENCES public.user(id_user),
  to_user INTEGER NOT NULL REFERENCES public.user(id_user),
  datasent TIMESTAMP NOT NULL,
  dataread TIMESTAMP NOT NULL,
  mes_content TEXT NOT NULL,
  attached_files INTEGER
);
exception when others then

  get stacked diagnostics
    v_state = returned_sqlstate,
    v_msg = message_text,
    v_detail = pg_exception_detail,
    v_hint = pg_exception_hint,
    v_context = pg_exception_context;

  raise notice E'Got exception:
    state : %
    message: %
    detail : %
    hint : %
    context: %', v_state, v_msg, v_detail, v_hint, v_context;

  raise notice E'Got exception:
    SQLSTATE: %
    SQLERRM: %', SQLSTATE, SQLERRM;

  raise notice '%', message_text; -- invalid. message_text is contextual to GET STACKED
  DIAGNOSTICS only

end; $$;

```

Та код рішення котре виводить дана функція:

```

NOTICE: Got exception:
  state : 42P07
  message: relation "messages" already exists
  detail :
  hint :
  context: SQL statement "create table messages (id_message SERIAL, from_user INTEGER,
to_user INTEGER, datasent TIMESTAMP, dataread TIMESTAMP, mes_content TEXT,
mes_content TEXT, attached_files INTEGER)"
PL/pgSQL function inline_code_block line 11 at SQL statement
NOTICE: Got exception:
  SQLSTATE: 42P07

```

					IC KP 122 HAI185 ПЗ	AP
						48
Зм.	Арк.	№ докум.	Підп.	ДА		

SQLERRM: relation "messages" already exists

ERROR: column "message\_text" does not exist

LINE 1: SELECT message\_text

^

QUERY: SELECT message\_text

CONTEXT: PL/pgSQL function inline\_code\_block line 33 at RAISE

SQL state: 42703

Дане середовище для управління реляційної бази даних допомагає услідкувати за багатьма недоліками, що можуть бути в результаті некоректної роботи компонентів програми.

## 4.2 Особливості конструювання програмних модулів

### 4.2.1 Особливості роботи з інтегрованим середовищем розробки

Середовище розробки – Visual Studio.

Microsoft Visual Studio — серія продуктів фірми Майкрософт, які включають інтегроване середовище розробки програмного забезпечення та низку інших інструментальних засобів. Ці продукти дозволяють розробляти як консольні програми, так і програми з графічним інтерфейсом, в тому числі з підтримкою технології Windows Forms, а також веб-сайти, веб-застосунки, веб-служби як в рідному, так і в керованому кодах для всіх платформ, що підтримуються Microsoft Windows, Windows Mobile, Windows Phone, Windows CE, .NET Framework, .NET Compact Framework та Microsoft Silverlight.

1. Visual Studio включає один або декілька з наступних компонентів:
2. Visual Basic .NET, а до його появи — Visual Basic
3. Visual C++
4. Visual C#
5. Visual F# (входить до складу Visual Studio 2010);
6. Visual Studio Debugger

### 4.2.2 Особливості створення програмної структури з урахуванням

					ІС КР 122 НАІ185 ПЗ	АР
						49
Зм.	Арк.	№ докум.	Підп.	ДА		

спеціалізованого Фреймворку.

Головний фреймворк розробки є ASP.NET Core.

Платформа ASP.NET Core представляє технологію від компанії Microsoft, призначену для створення різного роду веб-додатків: від невеликих веб-сайтів в великих веб-порталів і веб-сервісів.

ASP.NET Core може працювати поверх крос-платформної середовища .NET Core, котра в свою чергу може бути розгорнута на основних популярних операційних системах: Windows, Mac OS, Linux. І таким чином, за допомогою ASP.NET Core є можливість створювати крос-платформні додатки. І хоча Windows як середовище для розробки і розгортання програми досі превалює, але тепер вже ми не обмежені тільки цією операційною системою. Програми можливо запускати веб-додатки не тільки на ОС Windows, але і на Linux і Mac OS. А для розгортання веб-додатки можна використовувати традиційний IIS, або крос-платформний веб-сервер Kestrel.

#### 4.2.3 Особливості створення програмних класів

Програмні класи даного ПП будуть представлені в вигляді конструкторів, що приймають в собі значення get та set заданих параметрів атрибутів. Саме вони будуть приймати значення з БД.

Код створення програмних класів:

```
public class User
{
    public string login { get; set; }
    public string password { get; set; }
    public string e_mail { get; set; }
}

public class Administrator : User
```

```

{
    public int amount_of_commits { get; set; }
}

public class Event
{
    public string title { get; set; }
    public string beginning_time { get; set; }
    public bool state { get; set; }
}

public class Place
{
    public string name { get; set; }
    public string description { get; set; }
    public int placement { get; set; }
}

public class Information
{
    public string description { get; set; }
    public int assesment { get; set; }
    public string[] preferences = new string[10] { get; set; }
}

namespace PostgresApp
{
    public class ApplicationContext : DbContext
    {
        public DbSet<User> users { get; set; }
    }
}

```

```

public DbSet<Event> events { get; set; }

public DbSet<Information> hobbies { get; set; }

public ApplicationDbContext()
{
    Database.EnsureCreated();
}

protected override void OnConfiguring(DbContextOptionsBuilder
optionsBuilder)

{
    optionsBuilder.UseNpgsql("Host=localhost;Port=5432;Database=usersdb;Use
rname=postgres;Password=password");
}
}
}

```

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

Особливістю методів розробки алгоритмів на платформі ASP.NET Core слугує можливість конфігурації командами безпосередньо з бази даних. Як наслідок, завдяки цьому йде менше витрачання часу на виконання операцій, адже взаємодія йде напряду.

Код програми для метода userlogin:

```

public partial class userlogin: System.Web.UI.Page {
    string strcon =
    ConfigurationManager.ConnectionStrings["con"].ConnectionString;
    protected void Page_Load(object sender, EventArgs e) {
    }
    protected void Button1_Click(object sender, EventArgs e) {
    try {
        SqlConnection con = new SqlConnection(strcon);

```

```

if (con.State == ConnectionState.Closed) {
    con.Open();
}
SqlCommand cmd = new SqlCommand("select * from member_master_table
where member_id='" + TextBox1.Text.Trim() + "' AND password='" +
TextBox2.Text.Trim() + "'", con);
SqlDataReader dr = cmd.ExecuteReader();
if (dr.HasRows) {
    while (dr.Read()) {
        Response.Write("<script>alert('" + dr.GetValue(8).ToString() + "');</script>");
    } } else {
        Response.Write("<script>alert('Invalid credentials');</script>");
    } } catch (Exception ex) { } }

```

## 4.2 Модульне тестування програмних класів

### Перевірка функції Registration

Неправильне введення синтаксису реєстрації, як показано на рисунку

4.2.1 та наслідок на 4.2.2.

Рисунок 4.2.1 – Вхідні дані

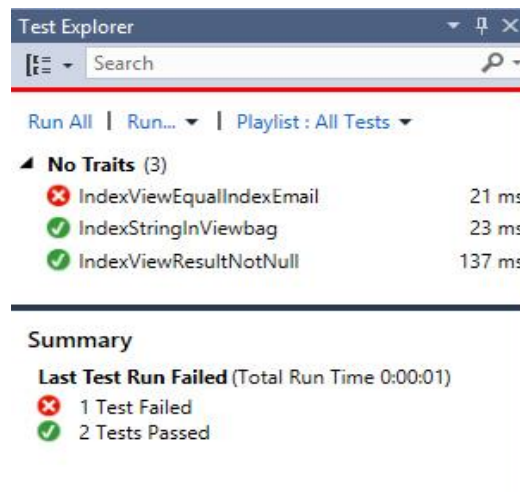


Рисунок 4.2.2 – Результат роботи тесту

## 5 РОЗГОРТАННЯ ТА ВАЛІДАЦІЯ ПРОГРАМНОГО ПРОДУКТУ

### 5.1 Інструкція з встановлення 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 допомогою маніпулятора «миша», якщо це комп'ютер чи сенсорного екрану зайти на сайт.

#### 5.2 Інструкція з використання програмного продукту

При заходженні на сайт – користувач потрапляє на головну сторінку, де в головному тілі знаходиться короткий опис даного додатку.

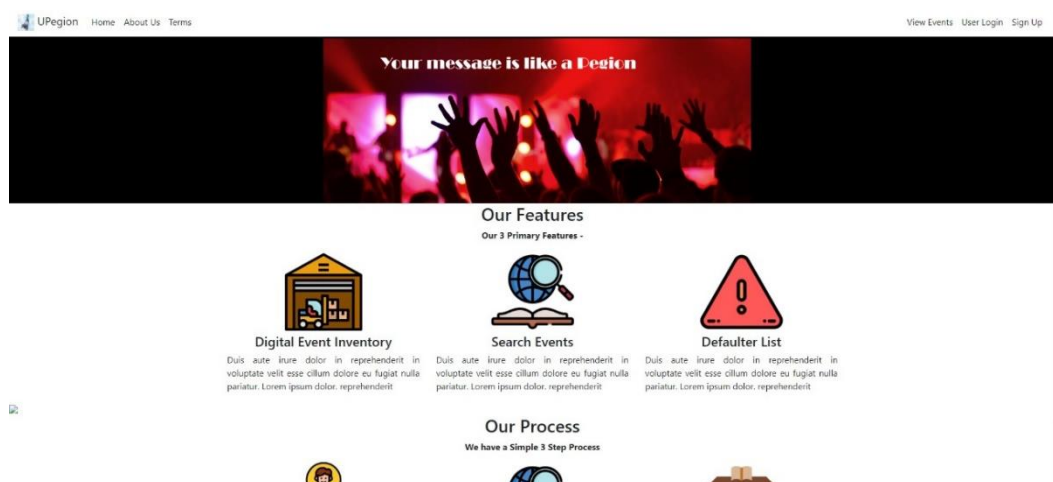


Рис 5.2.1 – Головна сторінка веб-додатку

Щоб отримати доступ до реєстрації, входження як користувач або задля ознайомлення з подіями - потрібно звернутися лівою кнопкою миші до посилань, що знаходяться в правому лівому кутку.

Якщо вибір пав на реєстрацію – з'явиться вікно реєстрації.

					ІС КР 122 НАІ185 ПЗ	АР
						55
Зм.	Арк.	№ докум.	Підп.	ДА		



Перехід від однієї сторінки до іншої здійснюється шляхом переходу по посиланням, що знаходяться зверху та знизу сайту, як показано на рисунку 5.2.2.

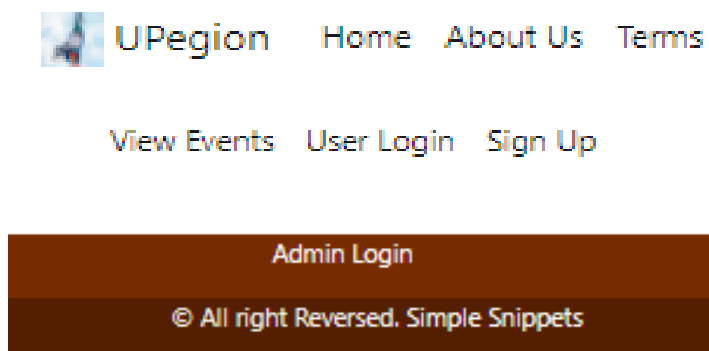


Рис 5.2.2 - Приклад способу пересування користувача між сторінками сайту

ПП надає можливість користувачу ролі «гість» вести параметри реєстрації (повне ім'я користувача, дата народження, ціль знайомства, пошта, регіон, місто, опис, пароль), як показано на рисунку 5.2.3.

Рис 5.2.3 - Приклад екранної форми реєстрації користувача

Для реєстрації користувач повинен ввести значення, як показано на рисунку 5.2.3.

					ІС КР 122 НАІ185 ПЗ	АР
						56
Зм.	Арк.	№ докум.	Підп.	ДА		

**Your Profile**  
Account Status - **Your status**

Full Name: Lida Date of Birth: 12/10/2020

Aim: To find a friend. Email ID: 2324

Region: Odessa City: Odessa Rating: 4

Your description: Lonely{

**Login Credentials**

User ID: User ID Old Password: Email ID New Password: Email ID

**Update**

<< Back to Home

Рис 5.2.4 - Приклад екранної форми заповнення даних реєстрації або їх зміни як користувача

### 5.3 Результати валідації програмного продукту

Метою програмного продукту є підвищення доступності різного роду заходів, тому ключовими класами даного додатку є користувач та події, що він створює та приймає участь.

**Member Login**

Member ID: guest1


Password: \*\*\*\*\*

**Login**

**Sign Up**


Рис 5.3.1 – Приклад входу в систему як користувача

## Event Details



Choose File No file chosen

**Event ID**

Member ID 

**Event Name**

Book Name

**Types**

Education ▼


Name

Whatdoyouwant ▼

**Creator Name**

Enter your name ▼

**Date**

mm/dd/yyyy 

**Street**

Armeiskaya  
Azovskaya  
Conductora  
Severnaya  
Molotova ▼

**Agess**

Enter age ban

**Any Cost?**

Event Cost(if it exists)

**Period**

Time

**Actual Stock**

Actual Stock

**Current Stock**

Current

**Last Event**

Pages

**Description**


Book Description

Add
Update
Delete

Рис 5.3.2 – Додавання події

[URegion](#) [Home](#) [About Us](#) [Terms](#)

[View Events](#) [User Login](#) [Sign Up](#)



The Admin Login form is centered on the page. It features a profile icon placeholder (a blue circle with a white silhouette and a yellow star) and the text "Admin Login" in bold. Below this, there are two input fields: "Admin ID" and "Password". A green "Login" button is positioned at the bottom of the form. A link "< Back to Home" is located below the form.


Рис 5.3.3 – Вхід Адміністратора

URegion

[Home](#)[About Us](#)[Terms](#)

[View Events](#)[User Login](#)[Sign Up](#)

Member Details



Member ID

Member ID

Full Name

Full Name

Account Status

Account Status

DOB

DOB

Contact No

Contact No

Email ID

Email ID

State

State

City

City

Pin Code

Pin Code

Full Postal Address

Full Postal Address

Delete User Permanently

Member List

<< Back to Home

Admin Login

© All right Reserved. Simple Snippets

Рис 5.3.4 – Регулювання даних про користувача на правах адміністратора

## ВИСНОВОК

В результаті створення програмного продукту була досягнута головна мета – збільшити рівень доступності для знаходження нових цікавих заходів та знайомство з новими людьми.

Як в наслідок цього – був створений веб-додаток, що повністю реалізує дану потребу. Гості можуть реєструватись як користувачі та організовувати власні заходи. Тим самим – вони розширюють свій круг знайомств.

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

- 1) проблема в організації обов’язків між членами команди;
- 2) проблема в відсутності дисципліни з приводу дотримання термінів роботи;
- 3) проблема відсутності досвіду роботи в команді;
- 4) 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 можливе реалізувати в майбутніх курсових роботах з урахуванням тем дисциплін наступних семестрів.

					ІС КР 122 НАІ185 ПЗ	АР
						59
Зм.	Арк.	№ докум.	Підп.	ДА		

## ПЕРЕЛІК ПОСИЛАНЬ

1. IT-ресурс «Metanit». [Електронний ресурс] – Режим доступу:  
<https://metanit.com/sharp/>
2. Документація бібліотеки Bootstrap [Електронний ресурс] – Режим доступу: <http://bootstrap-3.ru/index.php>
3. Документація PostgreSQL [Електронний ресурс] – Режим доступу:  
<https://www.postgresql.org/>
4. Вікіпедія [Електронний ресурс] – Режим доступу:  
[https://ru.wikipedia.org/wiki/Заглавная\\_страница](https://ru.wikipedia.org/wiki/Заглавная_страница)

					ІС КР 122 НАІ185 ПЗ	АР
						60
Зм.	Арк.	№ докум.	Підп.	ДА		