Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Кафедра прикладної математики

# ПОЯСНЮВАЛЬНА ЗАПИСКА

ДО КУРСОВОЇ РОБОТИ

з дисципліни “Бази даних та інформаційні системи”

на тему: Купівля квітів

Студентки IV курсу, групи КМ-42
напряму підготовки 6.040301 –
прикладна математика
БІЛОШАПКИ Ю.В.

Викладач
ТЕРЕЩЕНКО І.О.

Оцінка: ___ балів

Київ – 2017

# ЗАВДАННЯ НА ВИКОНАННЯ КУРСОВОЇ РОБОТИ

У даній курсовій роботі необхідно розробити систему, яка полегшуватиме процес купівлі та продажу квітів. Для реалізації даного завдання необхідно:

а) виконати передпроектне дослідження:

1) чітко визначити цілі даної системи;

2) застосувати технологію Scrum для планування та організації розробки даної системи;

3) визначити основні ролі, функції та бізнес – правила розроблюваного програмного забезпечення;

б) розробити архітектуру системи та побудувати основні діаграми:

1) визначити процеси кожного користувача в системі, побудувати діаграми переходів (Use cases) та діаграми послідовностей (Sequence diagram);
2) спроектувати логічну та фізичну структури баз даних, розробити та описати діаграми компонентів (Component diagram);
3) створення бази даних, виходячи з діаграм, згаданих у пункті 2;
4) визначити та передбачити перевірку на валідність даних системи;

в) розробка клієнтської та серверної частини системи, забезпечення взаємодії клієнт-серверу та написання API серверу.

# АНОТАЦІЯ

Тема даної курсової роботи – «Система продажу та купівлі квітів», в рамках якої було розроблено програмне забезпечення, яке полегшує та пришвидшує процес продажу квіткових товарів. За допомогою даної системи було вирішено проблему експрес замовлення та доставки даних замовлень з будь-якої точки та в будь-який час.

Дана робота складається з 5 розділів. Перший розділ містить передпроектне дослідження, де визначено основні цілі проекту, проведено аналіз існуючих систем. На підставі порівняльного аналізу розробляється система, яка зможе покращити та оптимізувати процес купівлі подарунків в порівнянні з існуючими системами.

У другому розділі сформульовані основні вимоги до розроблюваної системи, сформовано ділову модель, яка описує функціонування системи та визначає основні бізнес-правила системи.

У третьому розділі проведено моделювання та планування процесу розробки програмного застосунку, визначено основні компоненти системи та зв’язки між ними, визначено основні групи користувачів та їх ролі.

В четвертому розділі отримано семантичну модель даних, що відображає інформаційний зміст розроблюваної системи.

П’ятий розділ містить даталогічне проектування, в результаті якого отримуємо логічну та фізичну моделі системи.

# РЕФЕРАТ

Метою підготовки курсової роботи є закріплення знань, набутих при вивченні дисципліни «Бази даних та інформаційні системи», а основним завданням курсової роботи є проектування та розробка інформаційної системи для полегшення процесу купівлі подарунків.

Об’єктом дослідження курсової роботи є процес вибору, складання та покупки квіткових букетів.

Предметом дослідження є стадії відповідного процесу та етапи створення автоматизованої системи, яка буде реалізувати даний процес та допоможе оптимізувати його.

В даній курсовій роботі було проведено проектування потрібної системи та визначено етапи її реалізації, для чого було вивчено технології та методи планування розробки інформаційних систем. Крім того було проведено аналіз можливих груп користувачів системи, їхніх потреб, відповідно до цього були описані бізнес-правила функціонування системи та спроектовано необхідний функціонал.

Дана курсова робота присвячена плануванню та розробці інформаційних систем та проектуванню необхідної бази даних для функціонування цієї системи. Ключові слова: Scrum-планування, групи користувачів, Use Case діаграми, діаграма послідовностей, діаграма компонент, концептуальна модель системи, даталогічне проектування, логічна модель, фізична модель, база даних, сутність, атрибути, зв’язки між сутностями.

ЗМІСТ

# ВСТУП

Проблема швидкого та зручного доступу до товарів, їх віддалена покупка та доставка є досить актуальною та популярною у сучасному світі. Віддалені покупки та онлайн магазини є досить непоганою альтернативною ринковим відносинам та часто мають більше переваг, особливо коли це покупки з інших країн. Проблемою являється лише те, як би краще оптимізувати даний процес, які технології та ідеї впровадити задля отримання найкращого результату.

Сучасний стан інформаційних систем та технологій можна охарактеризувати наступними тенденціями:

1) наявність великої кількості промислово функціонуючих баз даних великого обсягу, що містять інформацію практично по всіх видах діяльності суспільства;

2) створення технологій, що забезпечують інтерактивний доступ масового користувача до цих інформаційних ресурсів;

3) розширення функціональних можливостей інформаційних систем, що забезпечують паралельну одночасну обробку баз даних з різноманітною структурою даних, мультиоб'єктних документів, гіперсередовища, в тому числі реалізують технології створення та ведення гіпертекстових баз даних.

Метою даної системи є оптимізація та пришвидшення продажу квітів. Дана область була обрана на підставі того, що купівля квітів часто буває рішенням спонтанним, і наявність відповідної системи, яка вміло виконає свою роль, користувалась б популярністю. На щастя, сьогодні інформаційні технології досить розвинені, за допомогою них можна створювати різні передові системи, які можуть виконувати людську роботу набагато швидше та краще. Тому постало питання про написання деякого інформаційного ресурсу, системи обслуговування з власними перевагами, яка забезпечить максимальною зручністю процес продажу та купівлі квітів.

# 2 ОСНОВНА ЧАСТИНА

## 1 АНАЛІЗ ПІДПРИЄМСТВА АВТОМАТИЗАЦІЇ

Вирішення проблеми спрощення процесу купівлі та продажу квітів вимагає детального аналізу необхідності автоматизації даного процесу. По-перше, розробка програмного забезпечення має базуватись на визначених принципах, правилах, які б в загальному описували роботу даного сервісу. Наприклад, можна забезпечити вибір товарів за допомогою фільтрації різними шляхами, зазвичай користувачі не хочуть переглядати і шукати необхідний їм зі всього об’єму товарів на складі. Також автоматизація може спростити процес оплати замовлення, наприклад, використовуючи банківські картки, хоча це також за бажанням замовника.  Але найпершою причиною, для чого автоматизація є необхідною – економія часу людей при виборі, замовленні та отриманні свого товару.

# 2 ПОСТАНОВКА ЗАДАЧІ

Для полегшення процесу продажу та купівлі квітів було обрано створити сайт інтернет-магазин продажу квітів. Метою створення такого сервісу є пришвидшення та полегшення вибору та продажу різних видів квітів, які можна обрати на сайті з можливістю складання власного букету, складові якого також мають бути обрані на сайті. Сервіс передбачає можливість електронної оплати карткою або ж оплату готівкою в національній валюті та доставки за вказаною адресою. Особливістю сайту являється те, що користувачу пропонуватимуться товари у вигляді рекомендацій, які випливають з попередніх замовлень та переглядів сторінок товарів даного користувача.

## 2.1 Граничні умови

Передбачено наступні граничні умови та обмеження сервісу:

1) товари доступні для перегляду всім гостям сайту, а можливість купівлі надається лише зареєстрованим користувачам;
2) товари, доступні для продажу, мають бути наявними на складі магазину.

## 2.2 Категорії користувачів системи

В розроблюваній системі передбачено наступні ролі: користувач – гість, користувач – адміністратор та користувач – кур’єр:

1) користувачу - гостю доступні наступні дії: перегляд всіх одиниць товару з можливістю переходу на особисту сторінку товару, де наявний повний опис товару ( код товару, тип, назва, ціна, статус, кількість одиниць, наявних на складі, розмір, коментар, кодове ім'я адміністратора, який додав товар і тд). Зареєстрованому користувачу надаються такі ж ролі, як звичайному гостю, але з можливістю додавання товарів у корзину з оформленням замовлення ( назви товарів та їх коди, кількість одиниць товару, загальна ціна, дані про самого користувача для доставки).Зареєстрований користувач також може замовити букет власного виготовлення (обирає квіти, які туди входитимуть та упакування);

2) роль адміністратора включає в себе наступні дії: додавання товару на сторінку товару разом з описом (назва, тип, ціна, статус, кількість одиниць, наявних на складі, коментар), виходячи з накладної, виданої поставщиком на папері, редагування товару, змінюючи статус товару (наявний, відсутній) та ціну, видалення товару, якщо товар більше не буде поставлятись у магазин, назначення кур'єра на відповідне замовлення. Система автоматизовано визначає, який з кур'єрів є вільним і, в залежності від цього, надає рекомендаційні вказівки адміністратору щодо назначення кур'єра. Якщо вільних кур'єрів на даний час немає, система назначає того, який має звільнитися найшвидше. Орієнтовний час доставки: від 40 хв. до 80 хв;
3) роль користувача - кур'єра: доставка товарів шляхом підтвердження у своєму особистому кабінеті взяття замовлення (номер замовлення). Вартість доставки залежить від вартості замовлення: замовлення до 200грн. - доставка 30 грн, від 200грн. - безкоштовно.

## 2.3 Основний функціонал та класи даних

а) додавання товару:

1) час виконання – діапазон < 1хв;

2) джерело інформації – накладна постачальника;

3) результатом виконання функції є збереження даних в базу даних доступних товарів.

б) створення власного букету:

1) час генерації букету, виходячи з вибраних одиниць – діапазон <10с;

2) ім’я користувача, який створив товар;

3) номер замовлення, в яке надійде даний товар;

4) коди та назви тих одиниць товару, які складають букет;

5) результатом виконання функції є внесення даних в базу даних замовлених товарів.

в) оформлення замовлення:

1) час виконання – діапазон < 1хв;

2) ім’я та код користувача, місто, район, вулиця, адреса будинку, поштовий індекс, номер телефону, який оформлює замовлення;

3) номер замовлення;

4) оплата через електронний гаманець, підтвердження оплати;

5) результатом виконання функції є інформація в кабінеті користувача про кур’єра (ім’я, номер телефону) та час доставки.

## 2.4 Бізнес – правила

Все, що стосується інформаційного забезпечення діяльності та являє собою процес або процедуру — бізнес-логіка; все, що не є процесом чи процедурою — бізнес - правило. Бізнес-логіка визначає, як бізнес-об'єкти взаємодіють один з одним та забезпечує маршрути і методи, за допомогою яких інформація про бізнес-об'єкти може бути доступною і оновленою. Бізнес-правила розглядаються як вимоги до моделей справжніх бізнес-об'єктів.

Для даної системи було визначено наступні основні бізнес – правила:

1) робити покупки, обирати товари у кошик можуть лише авторизовані користувачі системи, при відмові від замовлення після оплати протягом 40 хвилин – гроші повертаються у повному обсязі, більше 40 хвилин – повертається лише 80% заплаченої суми;
2) користувач не може додавати у замовлення більше ніж 10 одиниць товару, оскільки на одне замовлення надається лише один кур’єр і можуть бути складності з доставкою;
3) при затримці доставки більш ніж за 40 хв – доставка безкоштовно;
4) оплата здійснюється за бажанням користувача: онлайн через банківську картку або готівкою кур’єру, необхідно лише вказати спосіб на сторінці замовлення;
5) дані користувача можуть використовуватись для розсилок про оновлення на складі, акції, знижки і т.д. – таким чином користувачі завжди будуть в курсі та не втрачатимуть інтерес до сервісу.

# 3 МОДЕЛЮВАННЯ БІЗНЕС – ПРОЦЕСІВ

## 3.1 Scrum. Планування задач сервісу

Для реалізації системи продажу квітів було заплановано два спринти (sprints) – «Купівля квітів» та «Доставка». До першого спринту входять основні задачі сервісу: можливість вибору товару, додавання та редагування товарів, створення корзини з товарами, оформлення замовлення та створення власного букету. До другого спринту - задачі, які загалом обслуговують замовлення користувача.

До задачі «Додавання товару» (рис. 3.1) було додано вимоги до полів опису та завантаження зображення при додаванні нового товару на сайт (рис. 3.2).

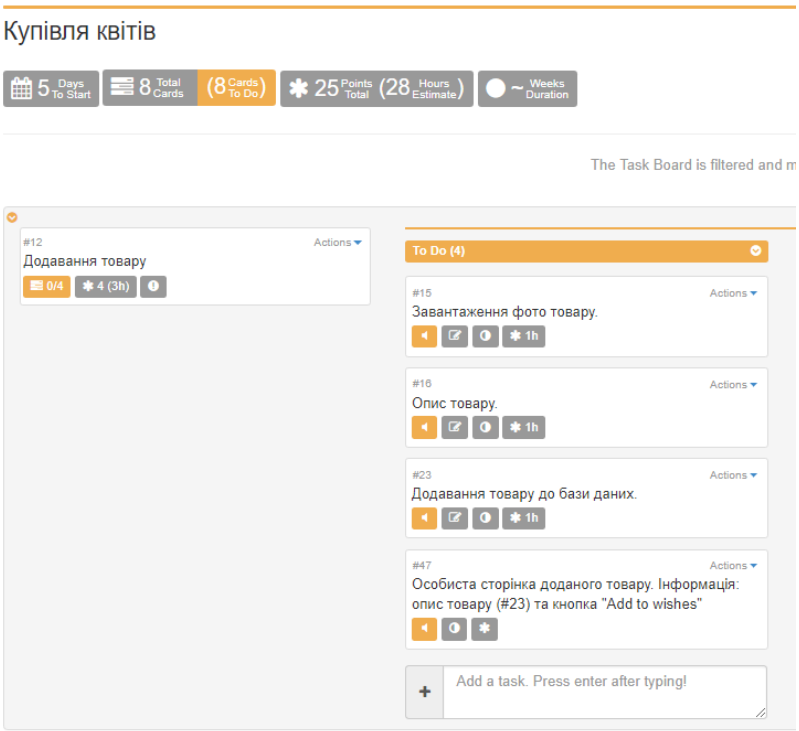


Рисунок 3.1 – Задача «Додавання товару»

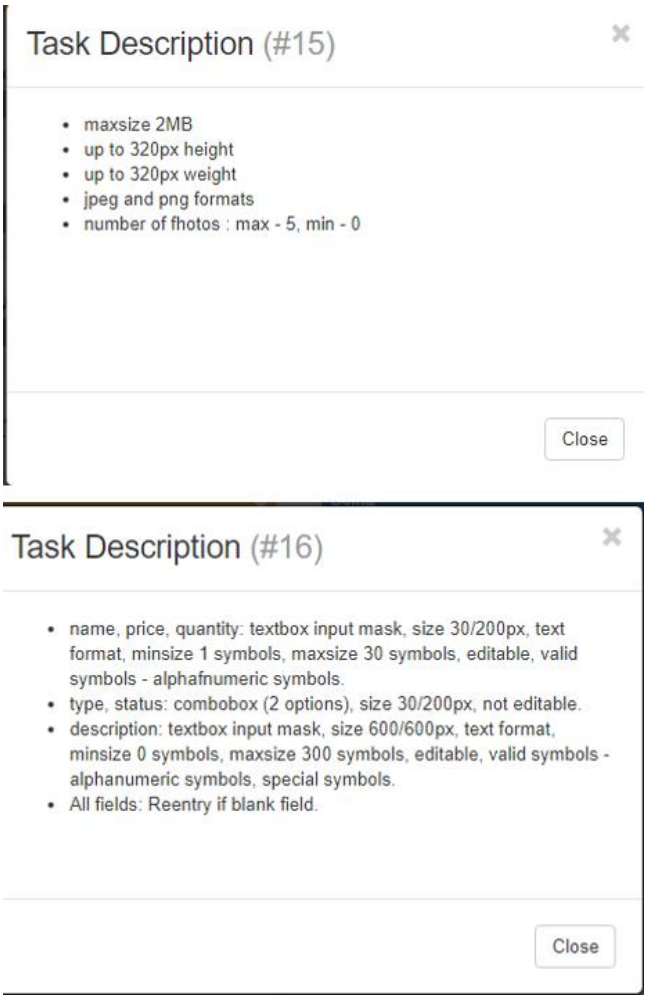


Рисунок 3.2 – Вимоги до полів опису товару та зображення товару

До задачі «Вибір товарів» було додано вимоги до полів фільтру для товарів та корзини (рис. 3.3).

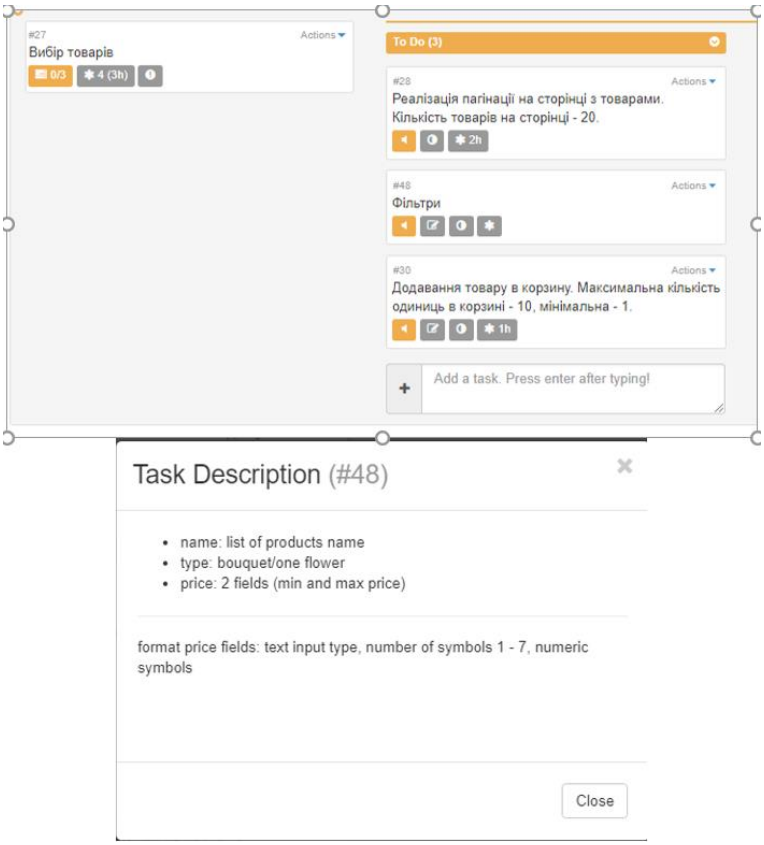


Рисунок 3.3 – Задача вибір товарів та вимоги до фільтрів

До задачі Авторизація було додано вимоги до полів введення користувачем даних для реєстрації на сайті (рис.3.4).

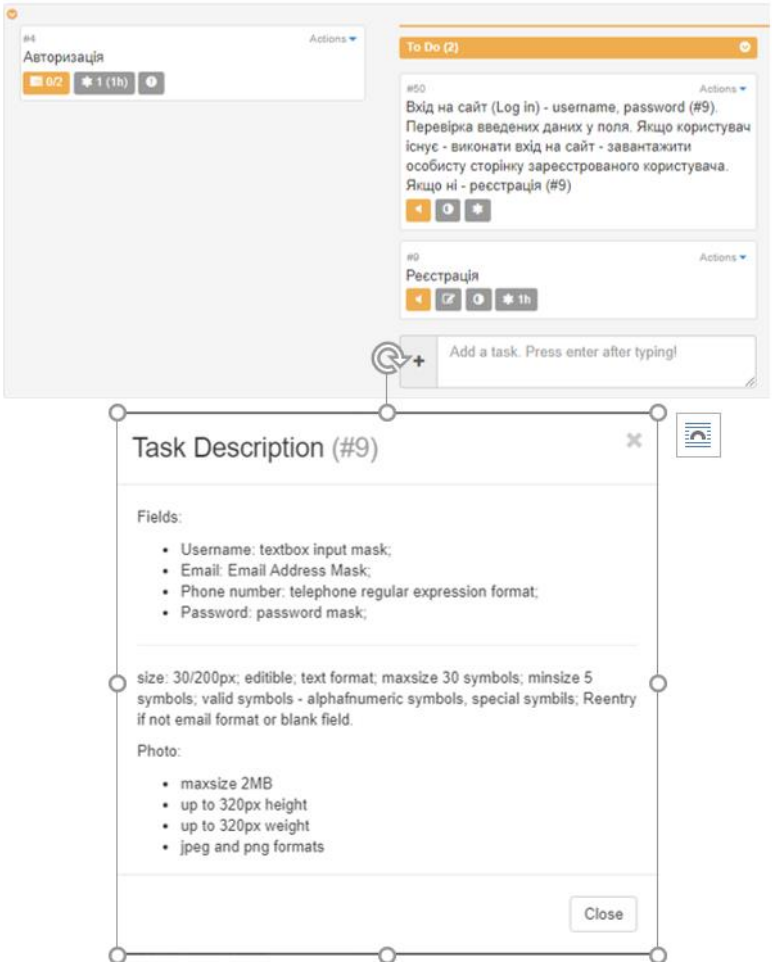


Рисунок 3.4 – Задача авторизації та вимоги до полів

До задачі «Оформлення замовлення» було додано вимоги до полів введення користувачем даних для доставки (рис.3.5).

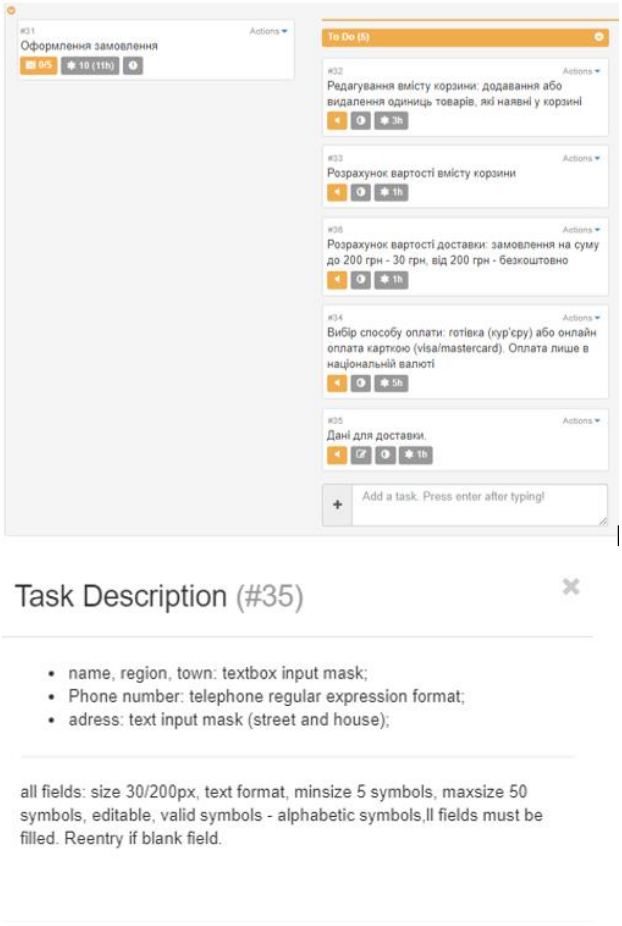


Рисунок 3.5 – Задача оформлення замовлення та вимоги до полів

## 3.2 Побудова Use Cases

Для опису поведінки та переходів між інтерфейсами сайту користувача було запропоновано наступну Use Case діаграму, яка зображена на рисунку 3.6. Зв'язок асоціації (association), позначений суцільною стрілкою означає, що актор - користувач, ініціює відповідний варіант використання сайту. Всі вершими діаграми - інтерфейси або процеси, що надають користувачу певний функціонал, а стрілки - переходи користувача між інтерфейсами та взаємозв'язок між інтерфейсами.

Вкладений перехід від одного інтерфейсу (процесу) до іншого інтерфейсу (процесу) означає, що другий є вкладеним підпроцесом першого (пуста стрілка). Вершина для вкладених переходів є деякою абстракцією, що реалізується через них, як, наприклад, інтерфейс «Product page» є абстракцією для вкладених, оскільки по своїй суті являє собою «List of flowers».

Розширений перехід від одного інтерфейсу (процесу) до іншого інтерфейсу (процесу) означає, що другий інтерфейс розширює функціонал першого, тобто для розширених переходів вершина існує як окремий функціонал (пунктирна стрілка). Наприклад, інтерфейси «Navigation» та « Filter» розширюють функціонал інтерфейсу «List of flowers», до того ж ці інтерфейси не можуть існувати без своєї вершини.

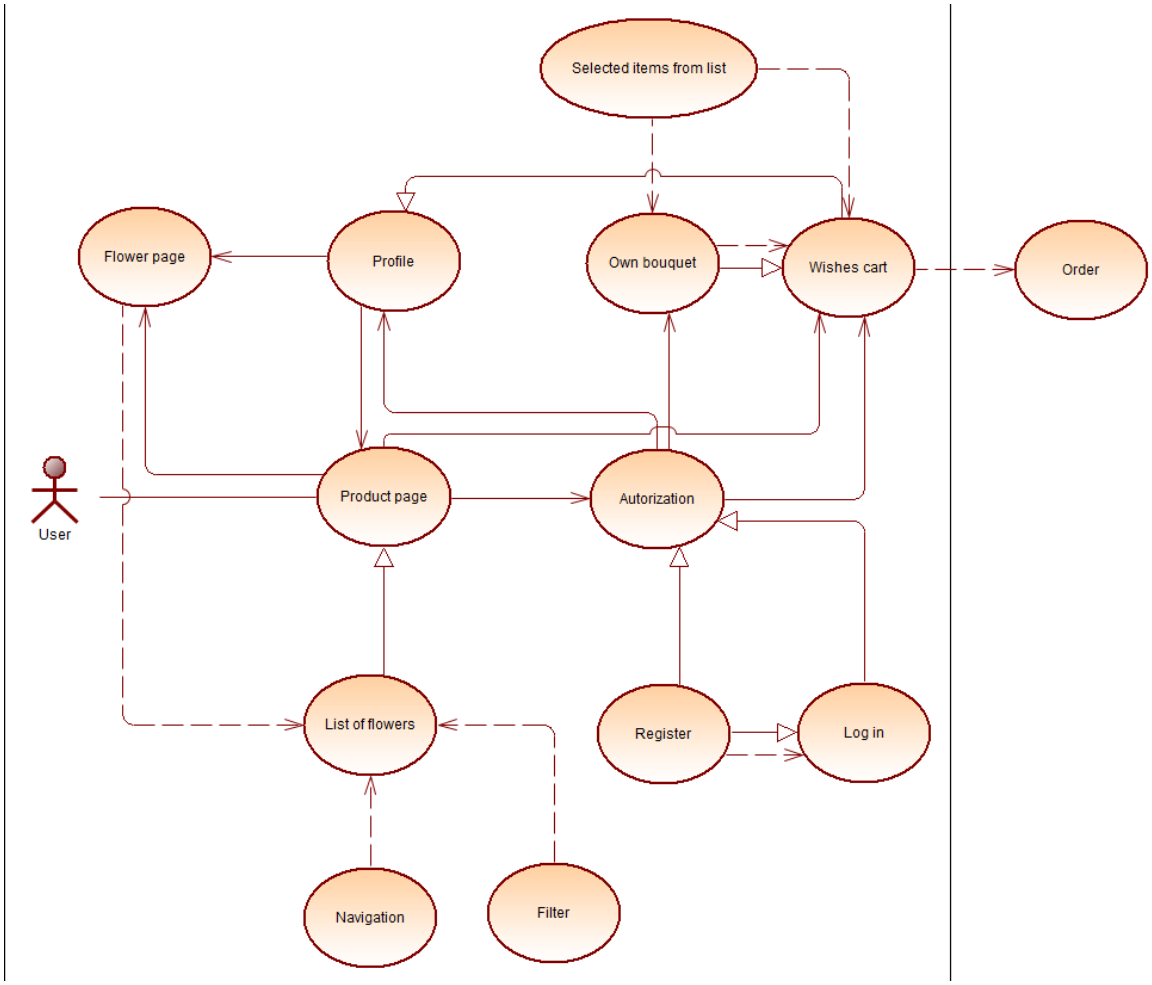


Рисунок 3.6 – Use case діаграма для користувача

Для опису поведінки та переходів між інтерфейсами сайту адміністратора було запропоновано наступну Use Case діаграму, яка зображена на рисунку 3.7. Відмінністю являється те, що адміністратор для входу у систему має спочатку авторизуватись, а також були додані додаткові процеси: додавання, редагування та видалення товарів.

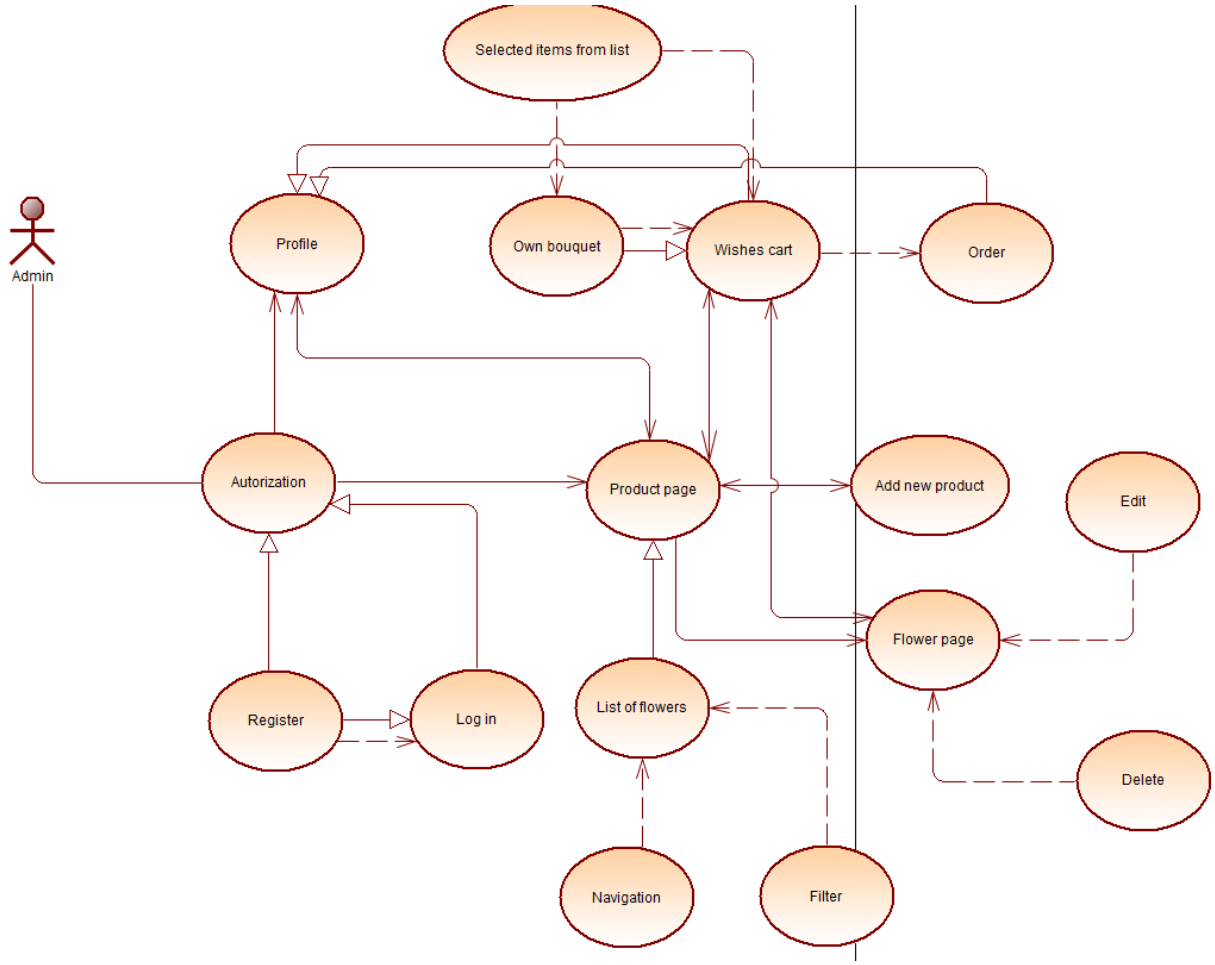


Рисунок 3.7 - Use case діаграма для адміністратора

## 3.3 Побудова компонентної діаграми (component diagram)

Для даної системи було визначено основні компоненти – блоки функціоналу та модулі: блок авторизації/реєстрації, модуль бази даних, блок покупки та блок товару. Між даними компонентами було описано основні зв’язки, функції та процедури, які регулюють основні процеси системи та які мають забезпечувати роботу.

# 4 ІНФОЛОГІЧНЕ ПРОЕКТУВАННЯ

## 4.1 Визначення та опис сутностей

У даній системі було визначено 4 основні сутності: користувач, роль якого визначається його атрибутами, квітка, яка являється одиницею товарів, та сукупність яких складає сутність замовлення, та сутність доставка.

Сутність користувач (user) складається з таких атрибутів:

- логін, являється ключовим атрибутом;
- повне ім’я користувача;
- email користувача;
- пароль користувача;
- номер телефону;
- тип користувача, являється зовнішнім ключем сутності «тип користувача»;
- фотографія – не являється обов’язковим атрибутом.

Сутність «квітка» складається з наступних атрибутів:

- назва, являється ключовим атрибутом;
- ціна за 1 шт.;
- статус товару;
- кількість даних одиниць на складі;
- коментар та фотографія – не являються обов’язковими атрибутами.

Сутність «замовлення» складається з таких атрибутів:

- час підтвердження замовлення, являється ключовим атрибутом;
- статус замовлення;

Сутність «доставка» складається з атрибутів:

- місто доставки;
- вулиця;
- будинок;
- номер квартири;
- ціна доставки (варіюється);

Всі сутності та зв’язки між ними описані в концептуальній діаграмі (conceptual ERD), зображеній на рисунку 4.1.

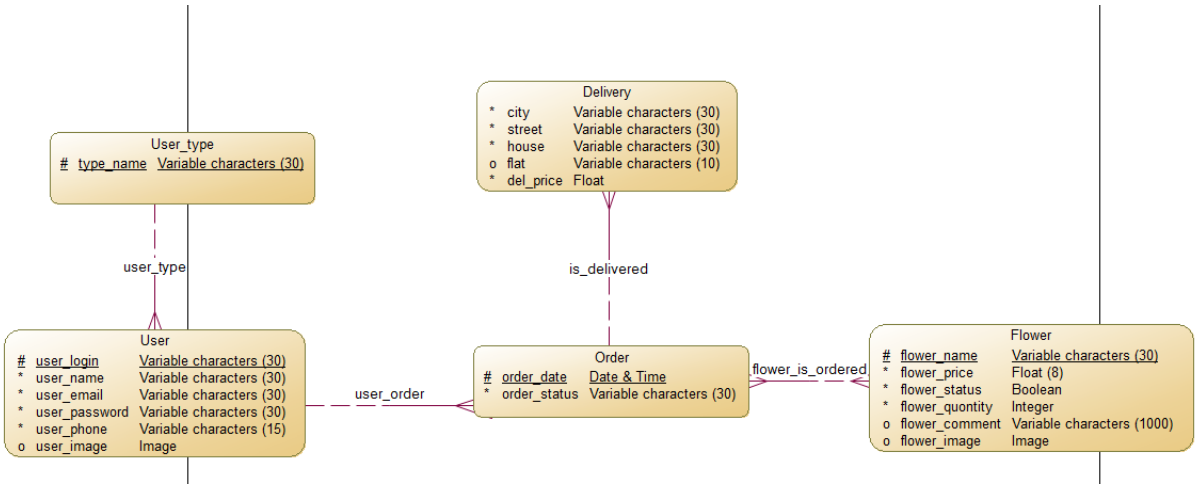


Рисунок 4.1 – Conceptual Data Model

# 5 ДАТАЛОГІЧНЕ ПРОЕКТУВАННЯ

## 5.1 Логічна модель даних

Логічна модель бази даних являється модифікацією концептуальної діаграми: в ній у кожній сутності з’являються додаткові атрибути, які є зовнішніми ключами інших сутностей, між якими встановлено зв’язок «залежність по ключу». Логічна модель даних зображена на рисунку 5.1.

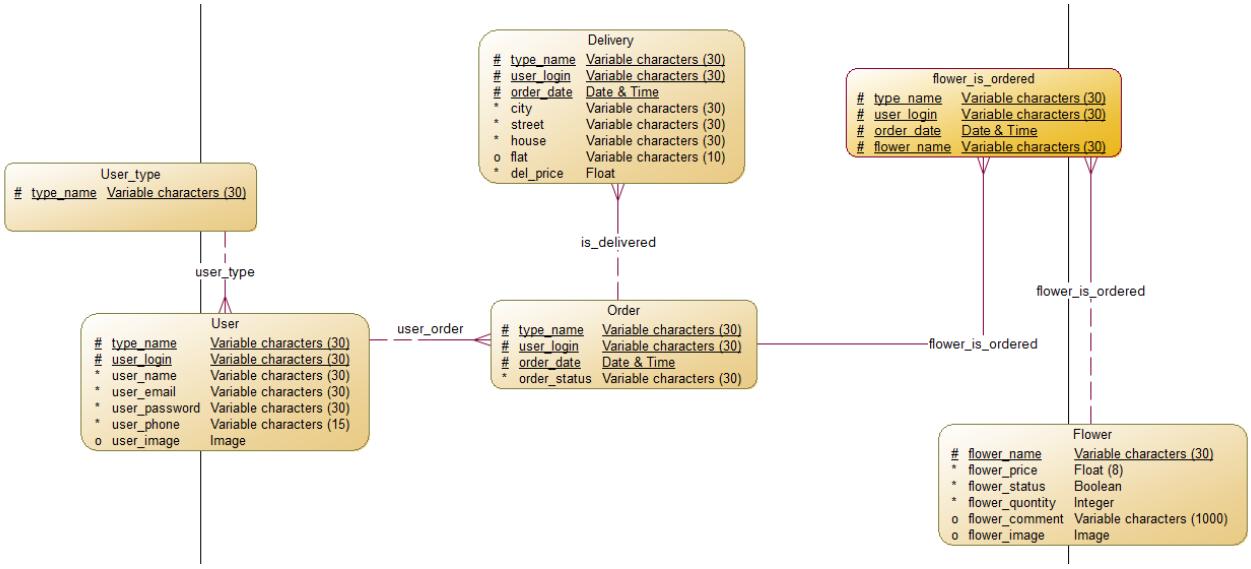


Рисунок 5.1 – Logical Data model

Логічну модуль даних було оптимізовано шляхом введення штучних primary key, щоб уникнути накопичення унікальних ключів у декількох сутностей. Оптимізована логічна модель даної бази даних зображена на рисунку 5.2.

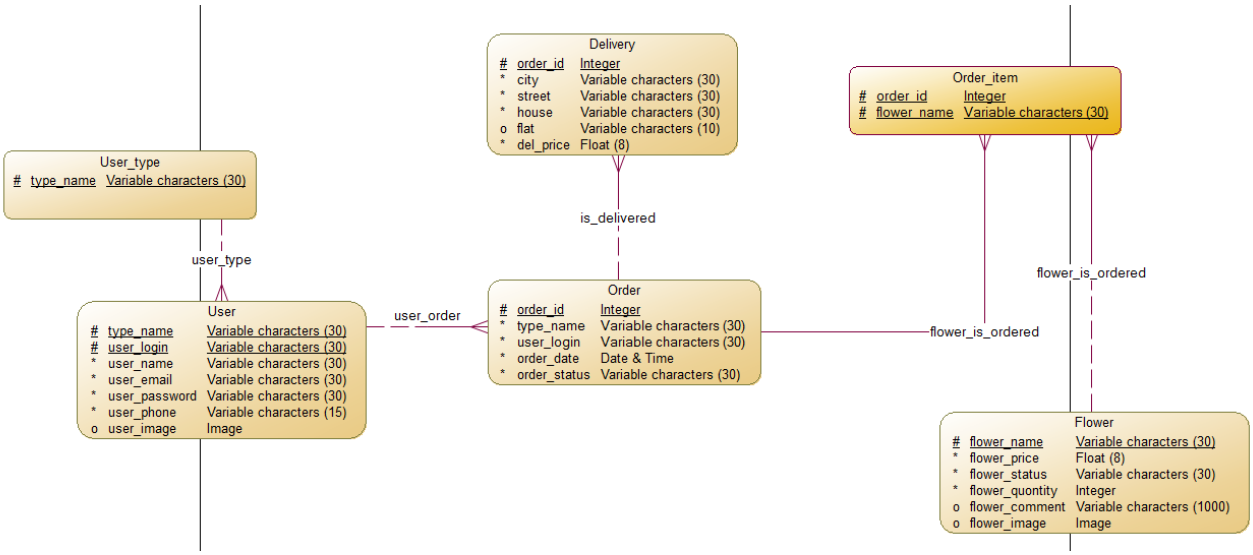


Рисунок 5.2 – Оптимізована Logical Data Model

## 5.2 Фізична модель бази даних

З логічної моделі даних автоматично було створено фізичну модель даних, для того щоб потім згенерувати кінцевий варіант бази даних у форматі SQL запитів. Фізична модель баз даних зображена на рисунку 5.3.

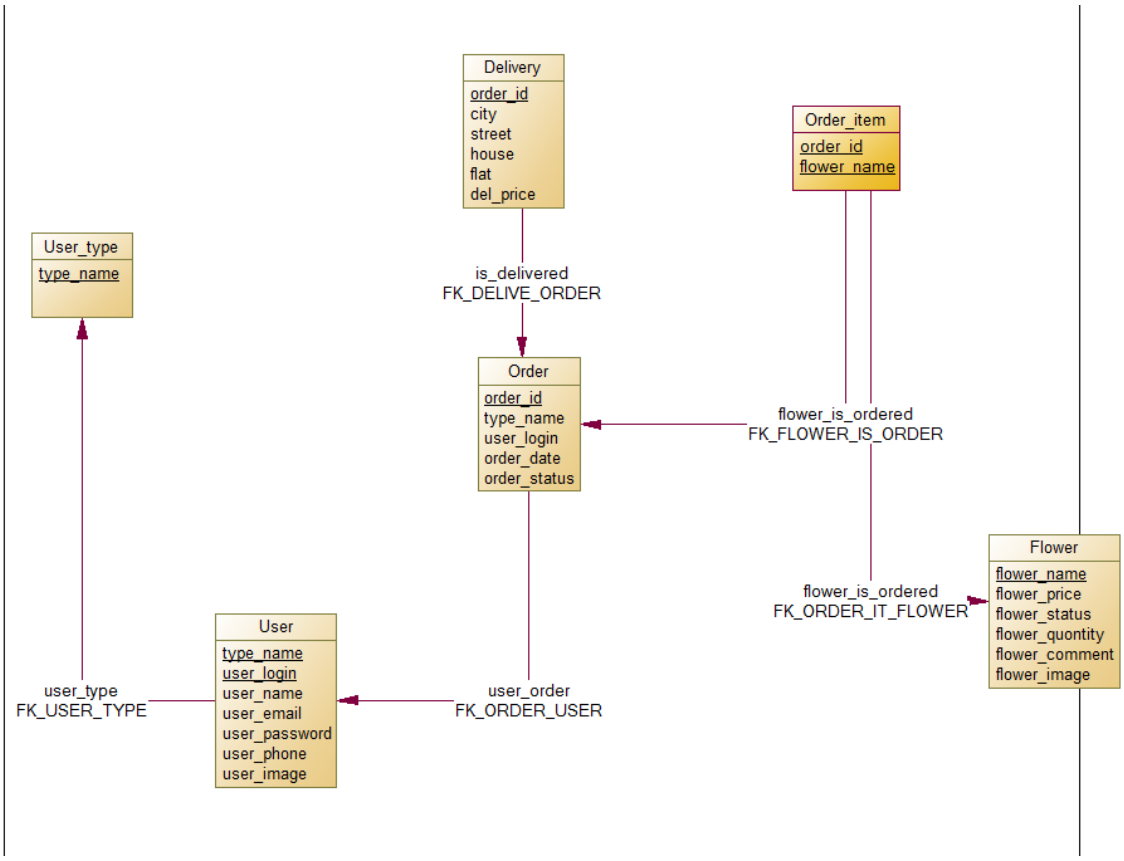


Рисунок 5.3 - Physical Data Model

# ВИСНОВКИ

У ході виконання курсової роботи було розроблено інформаційну систему, що має клієнт-серверну архітектуру,призначена для спрощення купівлі та продажу квітів. Проект отримує і зберігає дані у БД Oracle 11g. У БД міститься інформація про зареєстрованих користувачів, інформація про квітки, які продаються, а також можливість робити замовлення. У системі реалізовано дві ролі: користувач та адміністратор. Кожен користувач відповідно до своєї ролі може використовувати певний функціонал. При роботі з системою користувач може обирати квітки та оформлювати замовлення. Перевагами використання розроблюваної системи є те, що вона дозволяє користувачам економити час, який би вони затратили на фізичну купівлю квітів, та дозволяє зберігати всю важливу інформацію в одному місці не переживаючи про її втрату та мати до неї доступ у будь-який час.

# СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Форта Освой самостоятельно SQL. 10 минут на урок, 3-е издание.: Пер. С англ. – М.: Издательский дом «Вильямс», 2006 – 288 с.

2. Понимание SQL – Режим доступу http://www.sql.ru/docs/sql/u_sql/

3. SQL Tutorial – Режим доступу http://www.w3schools.com/sql/default.asp

# ДОДАТОК А. Лістинг файлу create.sql

```
/*==============================================================*/
/* DBMS name:      ORACLE Version 11g                           */
/* Created on:     15.11.2017 21:16:21                          */
/*==============================================================*/

alter table Delivery
   drop constraint FK_DELIVE_ORDER;

alter table "Order"
   drop constraint FK_ORDER_USER;

alter table Order_item
   drop constraint FK_FLOWER_IS_ORDER;

alter table Order_item
   drop constraint FK_ORDER_IT_FLOWER;

alter table "User"
   drop constraint FK_USER_TYPE;

drop table Delivery cascade constraints;

drop table Flower cascade constraints;

drop index user_order_FK;

drop table "Order" cascade constraints;

drop index flower_is_ordered_FK;

drop index flower_is_ordered2_FK;

drop table Order_item cascade constraints;

drop index user_type_FK;

drop table "User" cascade constraints;

drop table User_type cascade constraints;

/*==============================================================*/
/* Table: Delivery                                              */
/*==============================================================*/
create table Delivery
```

```
(
  order_id          INTEGER           not null,
  city              VARCHAR2(30)      not null,
  street            VARCHAR2(30)      not null,
  house             VARCHAR2(30)      not null,
  flat              VARCHAR2(10),
  del_price         FLOAT(8)          not null,
  constraint PK_DELIVERY primary key (order_id)
);

/*==============================================================*/
/* Table: Flower                                                */
/*==============================================================*/
create table Flower
(
  flower_name       VARCHAR2(30)      not null,
  flower_price      FLOAT(8)          not null,
  flower_status     VARCHAR2(30)      not null,
  flower_quontity   INTEGER           not null,
  flower_comment    VARCHAR2(1000),
  flower_image      VARCHAR2(256),
  constraint PK_FLOWER primary key (flower_name)
);

/*==============================================================*/
/* Table: "Order"                                               */
/*==============================================================*/
create table "Order"
(
  order_id          INTEGER           not null,
  type_name         VARCHAR2(30)      not null,
  user_login        VARCHAR2(30)      not null,
  order_date        DATE              not null,
  order_status      VARCHAR2(30)      not null,
  constraint PK_ORDER primary key (order_id)
);

/*==============================================================*/
/* Index: user_order_FK                                         */
/*==============================================================*/
create index user_order_FK on "Order" (
  type_name ASC,
  user_login ASC
);

/*==============================================================*/
```

```
/* Table: Order_item                                    */
/*==============================================================*/
create table Order_item
(
  order_id            INTEGER             not null,
  flower_name         VARCHAR2(30)        not null,
  constraint PK_ORDER_ITEM primary key (order_id, flower_name)
);

/*==============================================================*/
/* Index: flower_is_ordered2_FK                                 */
/*==============================================================*/
create index flower_is_ordered2_FK on Order_item (
  flower_name ASC
);

/*==============================================================*/
/* Index: flower_is_ordered_FK                                  */
/*==============================================================*/
create index flower_is_ordered_FK on Order_item (
  order_id ASC
);

/*==============================================================*/
/* Table: "User"                                                */
/*==============================================================*/
create table "User"
(
  type_name           VARCHAR2(30)        not null,
  user_login          VARCHAR2(30)        not null,
  user_name           VARCHAR2(30)        not null,
  user_email          VARCHAR2(30)        not null,
  user_password       VARCHAR2(30)        not null,
  user_phone          VARCHAR2(15)        not null,
  user_image          VARCHAR2(256),
  constraint PK_USER primary key (type_name, user_login)
);

/*==============================================================*/
/* Index: user_type_FK                                          */
/*==============================================================*/
create index user_type_FK on "User" (
  type_name ASC
);

/*==============================================================*/
```

```
/* Table: User_type                                */
/*==============================================================*/
create table User_type
(
  type_name           VARCHAR2(30)        not null,
  constraint PK_USER_TYPE primary key (type_name)
);

/*==============================================================*/
/* Table: Constraints                                */
/*==============================================================*/

ALTER TABLE "User"
        ADD CONSTRAINT user_check_name_regexp
        CHECK ( REGEXP_LIKE (user_name, '^[A-ZA-ЯІЇЄ][a-za-яіїє-]+($|\s)', 'c'));

ALTER TABLE "User"
        ADD CONSTRAINT user_check_name_len
        CHECK (length(user_name) <= 30);

ALTER TABLE "User"
        ADD CONSTRAINT user_check_login_regexp
        CHECK ( REGEXP_LIKE (user_login, '^[A-Za-z0-9.@#_]+($|\s)', 'c'));

ALTER TABLE "User"
        ADD CONSTRAINT user_check_login_len
        CHECK (length(user_login) <= 30);

ALTER TABLE "User"
        ADD CONSTRAINT user_check_email_regexp
        CHECK( REGEXP_LIKE (user_email, '^[A-Za-z!#$%&*+-=?^_`{|}~\.0-9]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,15}($|\s)'));

ALTER TABLE "User"
        ADD CONSTRAINT user_check_email_length
        CHECK(length(user_email) < 30);

ALTER TABLE "User"
        ADD CONSTRAINT user_check_password_regexp
        CHECK ( REGEXP_LIKE (user_password, '^[A-Za-z!#$%&*+-=?^_`{|}~\.0-9@]+($|\s)'));

ALTER TABLE "User"
        ADD CONSTRAINT user_check_password_len
        CHECK (length(user_password) >= 6 and length(user_password) <= 30);

ALTER TABLE "User"
        ADD CONSTRAINT user_check_phone_regexp
```

```
        CHECK ( REGEXP_LIKE (user_phone, '^[0-9]{2,15}($|\s)'));

ALTER TABLE "User"
        ADD CONSTRAINT user_check_phone_len
        CHECK (length(user_phone) > 2 and length(user_phone) <= 15);

ALTER TABLE "User"
        ADD CONSTRAINT user_check_image_regexp
        CHECK ( REGEXP_LIKE (user_image, '^[.0-9A-Za-z]{3,15}+\/+[0-9]{1,10}\/[.0-9a-zA-Z]{3,15}($|\s)'));

ALTER TABLE "User"
        ADD CONSTRAINT user_check_image_len
        CHECK (length(user_image) <= 256);

ALTER TABLE User_type
        ADD CONSTRAINT user_check_type_regexp
        CHECK ( REGEXP_LIKE (type_name, '^[A-Za-z0-9.@#]+($|\s)', 'c'));

ALTER TABLE User_type
        ADD CONSTRAINT user_check_type_len
        CHECK (length(type_name) <= 30);

ALTER TABLE Flower
        ADD CONSTRAINT flower_check_name_regexp
        CHECK ( REGEXP_LIKE (flower_name, '^[A-ZA-ЯІЇЄ][a-za-яіїє-]+($|\s)', 'c'));

ALTER TABLE Flower
        ADD CONSTRAINT flower_check_name_len
        CHECK (length(flower_name) <= 30);

ALTER TABLE Flower
        ADD CONSTRAINT flower_check_price_regexp
        CHECK ( REGEXP_LIKE (flower_price, '^[0-9]{1,4}.[0-9]{0,2}($|\s)'));

ALTER TABLE Flower
        ADD CONSTRAINT flower_check_status_regexp
        CHECK ( REGEXP_LIKE (flower_name, '^[A-Z][a-z]+($|\s)', 'c'));

ALTER TABLE Flower
        ADD CONSTRAINT flower_check_status_len
        CHECK (length(flower_name) <= 30);

ALTER TABLE Flower
        ADD CONSTRAINT check_user_quont_regexp
        CHECK ( REGEXP_LIKE (flower_quontity, '^[0-9]{1,10}($|\s)'));
```

```
ALTER TABLE Flower
        ADD CONSTRAINT flower_check_comment_regexp
        CHECK ( REGEXP_LIKE (flower_comment, '^[A-ZA-яІЇЄ][a-za-яіїє?.,! -]+($|\s)', 'c'));

ALTER TABLE Flower
        ADD CONSTRAINT flower_check_comment_len
        CHECK (length(flower_comment) <= 1000);

ALTER TABLE Flower
        ADD CONSTRAINT flower_check_image_regexp
        CHECK ( REGEXP_LIKE (flower_image, '^[.0-9A-Za-z]{3,15}+\/+[0-9]{1,10}\/[.0-9a-zA-Z]{3,15}($|\s)'));

ALTER TABLE Flower
        ADD CONSTRAINT flower_check_image_len
        CHECK (length(flower_image) <= 256);

ALTER TABLE "Order"
        ADD CONSTRAINT order_order_id_regexp
        CHECK ( REGEXP_LIKE (order_id, '^[0-9]{1,10}($|\s)'));

ALTER TABLE "Order"
        ADD CONSTRAINT order_check_status_regexp
        CHECK ( REGEXP_LIKE (order_status, '^[A-Z][a-z]+($|\s)', 'c'));

ALTER TABLE "Order"
        ADD CONSTRAINT order_check_status_len
        CHECK (length(order_status) <= 30);

ALTER TABLE Delivery
        ADD CONSTRAINT del_check_city_regexp
        CHECK ( REGEXP_LIKE (city, '^[A-ZA-яІЇЄ][a-za-яіїє-]+($|\s)', 'c'));

ALTER TABLE Delivery
        ADD CONSTRAINT del_check_city_len
        CHECK (length(city) <= 30);

ALTER TABLE Delivery
        ADD CONSTRAINT del_check_street_regexp
        CHECK ( REGEXP_LIKE (street, '^[A-ZA-яІЇЄ0-9][0-9a-za-яіїє-]+($|\s)', 'c'));

ALTER TABLE Delivery
        ADD CONSTRAINT del_check_street_len
        CHECK (length(street) <= 30);

ALTER TABLE Delivery
        ADD CONSTRAINT del_check_house_regexp
```

```
        CHECK ( REGEXP_LIKE (house, '^[0-9a-za-яіїє-]+($|\s)', 'c'));

ALTER TABLE Delivery
        ADD CONSTRAINT del_check_house_len
        CHECK (length(house) <= 30);

ALTER TABLE Delivery
        ADD CONSTRAINT del_check_price_regexp
        CHECK ( REGEXP_LIKE (del_price, '^[0-9]{1,4}.[0-9]{0,2}($|\s)'));

alter table Delivery
  add constraint FK_DELIVE_ORDER foreign key (order_id)
    references "Order" (order_id);

alter table "Order"
  add constraint FK_ORDER_USER foreign key (type_name, user_login)
    references "User" (type_name, user_login);

alter table Order_item
  add constraint FK_FLOWER_IS_ORDER foreign key (order_id)
    references "Order" (order_id);

alter table Order_item
  add constraint FK_ORDER_IT_FLOWER foreign key (flower_name)
    references Flower (flower_name);

alter table "User"
  add constraint FK_USER_TYPE foreign key (type_name)
    references User_type (type_name);
```