НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

(повне найменування інституту, факультету)

Автоматизованих систем обробки інформації і управління

(повна назва кафедри)

«До захисту допущено»
В.о. завідувача кафедри
Олександр ПАВЛОВ
(підпис) (ініціали, прізвище)
“ ” 2020 р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Програмне забезпечення інформаційних управляючих систем та технологій»

спеціальності «121 Інженерія програмного забезпечення»

на тему *Веб-застосування «Віртуальне ательє»*

Виконала: студентка IV курсу, групи *ІП-63 Корольова Людмила Вікторівна* _____
(прізвище, ім’я, по батькові) (підпис)

Керівник *Старший викладач Халус Олена Андріївна* _____
посада,науковий ступінь,вчене звання,прізвище,і ім’я, по батькові (підпис)

Консультант з графічної документації *доц., к.т.н., Ліщук К.І.* _____
посада,науковий ступінь,вчене звання,прізвище,і ім’я, по батькові (підпис)

Рецензент:

доц., к.т.н., доц. Петров П.П.
посада,науковий ступінь,вчене звання,прізвище,ім’я, по батькові (підпис)

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2020 року

Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет (інститут) *Інформатики та обчислювальної техніки*
(повна назва)

Кафедра *автоматизованих систем обробки інформації і управління*
(повна назва)

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – *121 Інженерія програмного забезпечення*

Освітньо-професійна програма – *Програмне забезпечення інформаційних управляючих систем та технологій*

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
_____ *Олександр ПАВЛОВ*
(підпис)
“___” _____2020 р.

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЄКТ СТУДЕНТУ

Корольовій Людмилі Вікторівні
(прізвище, ім’я, по батькові)

1. Тема проєкту «*Веб-застосування «Віртуальне ательє»*»

керівник проєкту *Халус Олена Андріївна, старший викладач*
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)
затверджені наказом по університету від “07” травня 2020 р. №1081-с

2. **Термін подання студентом проєкту** «08» *червня 2020 року*

3. Вихідні дані до проєкту

Технічне завдання

4. Зміст пояснювальної записки

1) Аналіз вимог до програмного забезпечення: основні визначення та терміни, опис предметного середовища, огляд існуючих технічних рішень та відомих програмних продуктів, розробка функціональних та нефункціональних вимог

2) Моделювання та конструювання програмного забезпечення: моделювання та аналіз програмного забезпечення, засоби розробки, технічні рішення, архітектура програмного забезпечення

3) Аналіз якості та тестування програмного забезпечення

4) Впровадження та супровід програмного забезпечення

5. Перелік графічного матеріалу

1) Схема структурна варіантів використань

2) Схема структурна бізнес-процесів

3) Схема бази даних

4) Схема структурна класів програмного забезпечення

5) Креслення вигляду екранних форм

6. Консультанти розділів проєкту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. **Дата видачі завдання** «10» *березня 2020 року*

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1.	*Вивчення рекомендованої літератури*	*19.03.2020*	
2.	*Аналіз існуючих методів розв'язання задачі*	*26.03.2020*	
3.	*Постановка та формалізація задачі*	*26.03.2020*	
4.	*Аналіз вимог до програмного забезпечення*	*02.04.2020*	
5.	*Алгоритмізація задачі*	*02.04.2020*	
6.	*Моделювання програмного забезпечення*	*09.04.2020*	
7.	*Обґрунтування використовуваних технічних засобів*	*16.04.2020*	
8.	*Розробка архітектури програмного забезпечення*	*23.04.2020*	
9.	*Розробка програмного забезпечення*	*30.04.2020*	
10.	*Налагодження програми*	*07.05.2020*	
11.	*Виконання графічних документів*	*14.05.2020*	
12.	*Оформлення пояснювальної записки*	*21.05.2020*	
13.	*Подання ДП на попередній захист*	*28.05.2020*	
14.	*Подання ДП рецензенту*	*03.05.2020*	
15.	*Подання ДП на основний захист*	*08.06.2020*	

Студент _____Людмила КОРОЛЬОВА
(підпис)

Керівник _____Олена ХАЛУС
(підпис)

№ з/п	Формат	Позначення	Найменування	Кількість листів	Примітка
1	А4		Завдання на дипломний проєкт	2	
2	А4	КПІ.ІП-6315.045440.02.81	Пояснювальна записка	62	
3	А4	КПІ.ІП-6315.045440.03.91	Технічне завдання	11	
4	А4	КПІ.ІП-6315.045440.04.51	Програма та методика тестування	6	
5	А4	КПІ.ІП-6315.045440.05.34	Керівництво користувача	11	
6	А4	КПІ.ІП-6315.045440.06.13	Опис програми	34	
7	А3	КПІ.ІП-6315.045440.07.99	Схема структурна варіантів використання	1	
8	А3	КПІ.ІП-6315.045440.07.99	Схема структурна бізнес-процесів	1	
9	А3	КПІ.ІП-6315.045440.07.99	Схема бази даних	1	
10	А3	КПІ.ІП-6315.045440.07.99	Схема структурна класів програмного забезпечення	1	
11	А3	КПІ.ІП-6315.045440.07.99	Креслення вигляду екранних форм	1	

Зм.	Арк.	ПІБ	Підп.	Дата	КПІ.ІП-6315.045440.00.90		
Розробн.		Корольова Л.В.			Відомість дипломного проєкту	Літ.	Лист 1 / Листів 1
Керівн.		Халус О.А.					
Консульт						КПІ ім.Ігоря Сікорського кафедра АСОІУ гр. ІП-63	
Н/контр.		Ліщук К.І.					
В.о.зав.каф		Павлов О.А.					

АНОТАЦІЯ

Пояснювальна записка дипломного проекту складається з чотирьох розділів, містить 44 таблиць та 10 джерел – загалом 124 сторінки.

Об’єкт дослідження: інтернет-магазин одягу та примірка одягу.

Мета дипломного проекту: розробити інтернет-магазин з онлайн-примірочною.

У першому розділі проведено аналіз відомих технічних рішень і розроблені вимоги до програмного.

У другому було розроблено архітектуру веб-застосування. Побудовано структурну схему класів та діаграму послідовності.

У третьому розділі проведено тестування інтернет-магазину за розробленим планом тестування. Описано процес тестування.

У четвертому розділі описано розгортання та впровадження веб-застосування.

У додатках наведено: опис програми, схема структурна класів програмного забезпечення, схема структурна послідовності виконання.

КЛЮЧОВІ СЛОВА: ПРИМІРКА, ІНТЕРНЕТ-МАГАЗИН, ОДЯГ, МАТЕРІАЛ ОДЯГУ, РОЗМІР ОДЯГУ

ABSTRACT

The explanatory note of the diploma project consists of four sections, contains 44 tables and 10 sources - a total of 124 pages.

Object of research: online clothing store and fitting of clothes.

The purpose of the diploma project: to develop an online store with an online fitting room.

In the first section the analysis of the known technical decisions is carried out and requirements to software are developed.

The second developed a web application architecture. The block diagram of classes and the sequence diagram are constructed.

In the third section, the online store was tested according to the developed testing plan. The testing process is described.

The fourth section describes the deployment and implementation of a web application.

The appendices contain: description of the program, scheme of structural classes of software, scheme of structural sequence of execution.

KEY WORDS: , ONLINE STORE, CLOTHING, CLOTHING MATERIAL, CLOTHING SIZE

Пояснювальна записка
до дипломного проєкту

на тему: Веб-застосування «Віртуальне ательє»

Київ – 2020 року

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

Інтернет-магазин – веб застосування для продажу одягу.

Онлайн-примірочна – сервіс що моделює зстилізоване зображення того, як одяг сидить на людині.

БД – база даних.

Адміністратор – людина, що авторизувалася у інтернет-магазмні як адміністратор.

Користувач – людина, що ввійшла у інтернет магазин.

Гість – людина, що ввійшла у інтернет магазин, але не є авторизованою.

Покупець – людина, що авторизувалася у інтернет-магазині як покупець.

Корзина – перелік товарів, які обрав покупець, проте не оформив замовлення.

ВСТУП

Одним із розповсюджених типів веб-застосувань є інтернет-магазин одягу. З одного боку такий магазин зручний тим що продавцям одягу немає необхідності утримувати фізичні примірочні: все що потрібно це склад, точка видачі одягу, якщо підтримується сомовивіз, та веб-застосування. Також це зручно і для покупців, оскільки вони можуть підібрати товар що їм подобається прямо з дому, не виникне такої проблеми що людина приїхала в одну точку продажу, а потрібний їй розмір залишився тількі в іншій що знаходиться у іншій частині міста[1].

Проте з іншого боку виникає велика проблема. Через відсутність фізичних примірочних людина точно не знає, як на ній буде сидіти обраний одяг. Така ситуація виникає з декількох причин. По-перше існує декілька систем розмірів[2], що залежить від країни виробника, по-друге розмір не дає знати, як саме буде сидіти модель одягу: одяг має різну довжину, може бути вільним або облягаючим і фото того як модель сидить на іншій людині не дає у повній мірі зрозуміти як саме ця модель буде сидіти на покупці. Через цю проблему в інтернет-магазини одяг повертають частіше ніж у звичайні магазини (в той час як у звичайних магазинів повернення товару складає приблизно 8% для інтернет-магазинів це приблизно 15% - 30%)[3].

Вирішенням даної проблеми є створення віртуального ательє яке дозволяє покупцю вказавши свої параметри побачити яким чином на ньому буде сидіти модель одягу. Для того щоб це ательє було точнішим за підбір одягу по розміру, покупець має вказати більшу кількість параметрів ніж враховується у системі розмірів, а продавець має вказати конкретні данні моделі одягу і його матеріал. У результаті враховуючи параметри покупця, параметри одягу та характеристики тканини з’являється можливість розрахувати як модель сидітиме на людині. Таке веб-застосування дозволить скоротити кількість повернень товару та збільшити довіру до магазину.

Проблема примірки одягу онлайн була висвітлена у статті[1].

Актуальність теми: зростання кількості інтернет-магазинів та їх актуалізація через єпідемію COVID-19 – у багатьох країнах світу звичайні магазини не мали можливості працювати через введеня карантину, в той час як для інтернет-магазинів така ситуаціє не заважає працювати.

Мета розробки: створення такого веб-застосування для продажу одягу, який буде на основі параметрів покупця та одягу будувати модель того як одяг сидить на покупці.

Завдання розробки:

- створення сервісу що на основі характеристик одягу генерує стилізоване зображення одягу та надає можливість перефарбувати стилізоване зображення в інший колір;
- створення сервісу що на основі вимірів людини генерує стилізоване зображення людини з відповідними пропорціями та надягає на стилізоване зображення людини стилізоване зображення одягу;
- веб-застосування для продажу одягу з інтеграцією цих сервісів;
- інтеграція з існуючою базою даних.

Практичне значення одержаних результатів: розроблене веб-застосування дозволить вказавши параметри покупця змоделювати як на ньому буде виглядати одяг.

1 АНАЛІЗ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Загальні положення

Для створення якісного програмного забезпечення необхідно чітко виділити вимоги, яким воно має відповідати та сформулювати на їх основі технічне завдання. За допомогою вимог описується продукт який буде відповідати потребам користувача і буде для користувача корисним.

Успішність інтернет-магазину напряму залежить від веб-застосування яким він користується. Тому дуже важливим є зібрати вимоги для розуміння яким саме власник хоче бачити магазин, яким функціоналом він має володіти і як виглядати. Це дозволить досягти цілей поставлених замовником і в результаті розроробки надати йому продукт який він хотів.

Змістовний опис і аналіз предметної області

Інтернет-магазин складається з фізичної частини – це склади, товари, матеріали та персонал, та з програмної частини. Програмна частина складається з веб-застосування для продажу товарів і з застосування для обліку товарів. Застосування для продажу повинно показувати наявні товари за різними категоріями, наприклад одяг, парфуми чи акссесуари, та фільтрувати іх за різними параметрами, наприклад за ціною та новизною. Також воно повинно давати можливість власне купувати один або декілька товарів. Інформація про кількість товарів та їх параметри береться з застосування для обліку товарів. Ці дві частини можуть бути як суміщенні так і розділені, і коли вони розділені з’являється також задача інтеграції програми обліку товарів з програмою продажу.

Додаткові задачі з’являються якщо це не звичайний магазин який купує і перепродає товари, а ательє, яке саме виготовляє одяг. Ательє закупає матеріали для одягу, шиє за своїми стандартними викройками або на

замовлення та продає виготовлений одяг. Звичайний магазин може бути суміщений з ательє.

Також для інтернет-магазинів типовими є системи знижок як на окремі товари, так і загалом для покупців які вже здійснили багато покупок у цьому магазині. Ця інформація зберігається у програмі обліку та може коригуватись адміністратором.

Якщо магазин одягу включає в себе як і звичайні магазини так і інтернет-магазин, то також з’являться задача синхронизації данних з точок продажу одягу і з інтернет-магазину

Аналіз успішних IT-проектів

Аналіз відомих технічних рішень

Структуру веб-застосування можна умовно поділити на такі частини: зберігання даних, шар роботи з даними, шар бізнес логіки та шар представлення, для кожно з цих шарів слід розглядати різні технічні рішення.

Найнижчим шаром можна важжати зберігання даних, а данні зазвичай зберігаються у базах данних. Для зберігання даних інтернет-магазину підходить реляційна база даних. База даних обиралася серед систем з відкритим кодом, оскільки платні системи не мають переваг принципових для веб-застосування з продажу одягу. Серед відкритих систем однією з найпопулярніших є PostgreSQL[6], оскільки вона більш гнучко працює з данними.

Далі йде шар роботи з даними DAL(Data Access Layer). Для реалізації цього шару зазвичай використовують одну з об’єктно-орієнтованих мов програмування таких як Java або C# на платформі .NET. Для даного веб-застосування було обрану мову програмування Java. Для роботи з данними було використано комбінацію JDBC та JPA, що є типоми для роботи з базою даних на Java.

Для шару бізнес-логіки BL(Business Logic) було використано комбінацію з двох мов програмування. Java було використано для реалізації обробки даних

з бази для надання їх на шар представлення, наприклад сортування та фільтрація товарів та для обробки інформації що надійшла з шару представлення. Для цієї мети було застосовано фреймоворк Spring.

Вирішення задачі поставленої в даної роботи передбачає інтелектуальну обробку даних. Даний аналіз даних не є складним, тому немає необхідності використовувати не класичні інструменти автоматиматизації процесів.

На даний момент вже існує достатня кількість бібліотек та фреймворків, що забезпечують полегшення створення програмного забезпечення. А з часом їх кількість буде рости. Що стосується аналізу даних, класичними інструментами є MATLAB, R, Python, Octave. В даній роботі був використано мову програмування Python.

MATLAB - це обчислювальне середовище з численними парадигмами та власна мова програмування, розроблена MathWorks. MATLAB дозволяє проводити матричні маніпуляції, побудову графіків функцій та даних, реалізацію алгоритмів, створення інтерфейсів користувача та взаємодію з програмами, написаними іншими мовами.

R - вільне програмне середовище для статистичних обчислень та графіки.

Octave - це програмне забезпечення, що містить мову програмування високого рівня, головним чином призначену для чисельних, математичних обчислень обчислень та аналызу даних. Octave допомагає в чисельному вирішенні лінійних і нелінійних задач, а також для виконання інших чисельних експериментів, використовуючи мову, яка здебільшого сумісна з MATLAB. Перевагою Octave, э те, що він безкоштовний.

Python – мова програмування з великою кількістю бібліотек реалізованих для обробки даних. Особливістю Python є застосування векторних обчислень що сильно прискорює обробку даних. Для даного додатку було застосовано такі бібліотеки як numpy, math та scikit-learn. Numpy – це бібліотека для зручної роботи з даними, яка дозволяє зберігати їх в масиві та робити іх перетворення та статистичні обрахунки над ними. Math – це бібліотека стандартних

математичних функцій (sin, cos, ln і т.д.) та констант (e, pi і т.д.). Scikit-learn – це бібліотека для інтелектуальної обробки даних, наприклад для побудови регресійних моделей чи кластерезаціїї. Уданій роботі використовується її модуль scikit-image для роботи із зображеннями.

Для шару представлення PL(Presentation Layer) зазвичай використовується мова програмування JavaScript. Є доволі багато різних фреймворків.

AngularJS - це фреймворк з відкритим вихідним кодом для створення фронтенда веб-додатків. Він націлений на вирішення завдань, з якими стикається розробник при побудові односторінкових додатків.

Цей фреймворк спрощує розробку і тестування додатків. Він реалізує підхід Model-View-Controller (MVC) і Model-View-ViewModel (MVVM).

React - це бібліотека JavaScript. Він використовується для побудови призначених для користувача інтерфейсів. React підтримується Facebook, Instagram та іншими компаніями. У шаблоні Model-View-Controller (MVC) він відповідає Поданню (View).

Для даної роботи було обрано AngularJS.

1.3.2 Аналіз відомих програмних продуктів

Існує дуже багато інтернет-магазинів, проте у дуже малої кількості з них є функція примірки. Типовими представниками є styleclub [4] та onvolga [5].

Styleclub - онлайн примірка з використанням стилізованих зображень одежі та стилізованого зображення людини. Перевагою є коректне відображення одягу, що вже дозволяє оцінити, як речі пасують одна одній. Недоліком є відсутність врахування параметрів людини.

Onvolga - Онлайн примірка з використанням фото реальних моделей та реальних фото одягу. Примірка відбувається вручну: користувач обирає одяг і розтягує його по розміру моделі. Перевагою є реальний вигляд одягу, недоліки

полягають в тому що одяг не підходить до заготовленої моделі та не враховуються параметри реального користувача.

В обох примірочних існує можливість одягати як одну так і декілька речей.

Також в деяких джерелах [2] згадуються примірки, які або примірюють одяг у доповненій реальності, або створюють детальну 3d-модель людини та одягають на неї одяг. Проте інтернет-магазинів з імплементованими примірочними такого типу не було знайдено.

Деякі інтернет-магазини вказують, що моють онлайн-примірку, проте насправді дана можливість є відсутньою, або реалізована для дуже малої кількості товарів з асортименту магазину.

1.4 Аналіз вимог до програмного забезпечення

Система повинна містити такі типи користувачів як адміністратор, гість та покупець.

Адміністратор – це користувач, який може управляти даними про товари та про користувачів.

Гість – це користувач, який може переглядати товари та створити аккаунт покупця.

Покупець – це користувач, якому доступні можливості перегляду, купівлі та примірки товарів.

В системі повинні бути реалізовані наступні функції:

- перегляд товарів за категоріями;
- сортування товарів за ціною;
- сортування товарів за новизною;
- пошук товарів за назвою;
- фільтрація товарів за ціною;
- примірка товарів;
- додання товарів у корзину;

– купівля товарів;
– управління товарами.

1.4.1 Розроблення функціональних вимог

Схема структурна варіантів використання наведена у графічному матеріалі.

В системі передбачено наступні варіанти використання, які описані у таблицях 1.1 – 1.29:

Таблиця 1.1 - Варіант використання UC001

Назва	Додавання товарів з програми обліку
Опис	Адміністратор може додавати товари з файлу.
Учасники	Адміністратор.

Таблиця 1.2 - Варіант використання UC002

Назва	Оновлення кількості товарів
Опис	Адміністратор може вручну змінювати кількість товарів.
Учасники	Адміністратор.
Передумови	Адміністратор виявив бракований товар чи товар було продано не через інтернет-магазин.
Постумови	В системі актуальна інформація про кількість товару.
Основний сценарій	Адміністратор вручну змінює кількість товару в системі.

Таблиця 1.3 - Варіант використання UC003

Назва	Ведення товарів
Опис	Адміністратор може оновити характеристики товару. з файлу
Учасники	Адміністратор.
Передумови	Інформація про товари застаріла.
Постумови	Характеристики товарів оновлені.
Основний сценарій	Адміністратор завантажує файл з новими характеристиками товарів у систему і характеристики товарів оновлюються.

Таблиця 1.4 - Варіант використання UC004

Назва	Ведення покупців
Опис	Адміністратор може вручну додавати нових покупців у систему.
Учасники	Адміністратор.
Передумови	Контакти покупця були збережені у сторонньому застосуванні, або покупець звернувся на пряму до адміністратора.
Постумови	В системі створено аккаунт покупця.
Основний сценарій	Адміністратор вручну створює аккаунт покупця.

Таблиця 1.5 - Варіант використання UC005

Назва	Ведення назв та цін товарів
Опис	Адміністратор може коригувати характеристики конкретних єкземплярів (наприклад товар конкретного розміру)

Продовження таблиці 1.5

Учасники	Адміністратор.
Передумови	Одна модель має різні характеристики в залежності від розміру
Постумови	Кожному єкземпляру відповідають свої характеристики
Основний сценарій	Адміністратор знаходить конкретний єкземпляр товару і змінює його характеристики

Таблиця 1.6 - Варіант використання UC006

Назва	Описання матеріалу розміру та кольору товару
Опис	Адміністратор може вносити характеристики товару необхідні для примірки
Учасники	Адміністратор.
Передумови	До магазину додано товар, який можна приміряти
Постумови	Для товару стає доступною функція онлайн-примірки
Основний сценарій	Адміністратор вносить данні про товар та відмічає його як такий, що можна приміряти

Таблиця 1.7 - Варіант використання UC007

Назва	Перегляд товарів у категорії
Опис	Користувач обирає категорію товарів, сортування та фільтрацію і переглядає товари.
Учасники	Гість або покупець.
Передумови	Користувач обрав категорію товарів.
Постумови	Відображено товари які відповідають параметрам.
Основний сценарій	Користувач обирає категорію, сортування та натискає пошук.

Таблиця 1.8 - Варіант використання UC008

Назва	Перегляд конкретного товару
Опис	Користувач переглядає товар, його конкретні розміри.
Учасники	Гість або покупець.
Передумови	Користувач обрав товар.
Постумови	На сторінці відображено товари які відповідають заданим параметрам.
Основний сценарій	Користувач обирає товар, натискає переглянути і бачить інформацію про товар.

Таблиця 1.9 - Варіант використання UC009

Назва	Перегляд категорій товарів
Опис	Користувач переглядає категорії товарів
Учасники	Гість або покупець.
Передумови	Користувач перейшов на головну сторінку
Постумови	На сторінці відображено категорії товарів
Основний сценарій	Користувач переходить на головну сторінку і бачить категорії товарів.

Таблиця 1.10 - Варіант використання UC010

Назва	Створення аккаунту покупця
Опис	Гість вводить свої данні та створює аккаунт покупця.
Учасники	Гість.
Передумови	Користувач зайшов у веб-застосування як гість.
Постумови	У систему додано новий аккаунт покупця.
Основний сценарій	Гість натискає на кнопку створити аккаунт, вносить свої данні та натискає на кнопку створити аккаунт.

Таблиця 1.11 - Варіант використання UC011

Назва	Вхід у аккаунт покупця
Опис	Гість перезаходиить у систему як покупець.
Учасники	Гість.
Передумови	Користувач має аккаунт покупця проте зайшов у систему як гість.
Постумови	Користувач зайшов у систему як гість.
Основний сценарій	Гість натискає на кнопку зайти, вводить свою пошту і пароль та заходить у систему як покупець.

Таблиця 1.12 - Варіант використання UC012

Назва	Внесення параметрів покупця
Опис	Покупець вносить свої параметри в систему.
Учасники	Покупець.
Передумови	Користувач ввійшов у систему як покупець та зайшов на власну сторінку.
Постумови	В систему додано параметри користувача.
Основний сценарій	Покупець заходить на власну сторінку та вносить свої параметри, зберігає їхі вони додаються у систему.

Таблиця 1.13 - Варіант використання UC013

Назва	Примірка
Опис	Покупець обирає одяг та примірює його
Учасники	Покупець.
Передумови	Покупець вніс данні про свої характеристики та обрав одяг який хоче приміряти.

Продовження таблиці 1.13

Постумови	Сгенеровано модель того як одяг сидітиме на покупці та виведено відповідне зображення.
Основний сценарій	Покупець заходить на сторінку примірки та обирає одяг що хоче приміряти, натискає примірати і бачить результат.

Таблиця 1.14 - Варіант використання UC014

Назва	Додавання до корзини
Опис	Покупець додає до корзини товар який планує купувати.
Учасники	Покупець.
Передумови	Покупець перейшов на сторінку товару.
Постумови	Покупець має можливість купити товар.
Основний сценарій	Покупець натискає на кнопку додати у корзину і товар додається до корзини.

Таблиця 1.15 - Варіант використання UC015

Назва	Купівля товарів
Опис	Покупець купує товари з корзини.
Учасники	Покупець.
Передумови	Товари додано до корзини покупця
Постумови	Створено замовлення яке чекає підтвердження.
Основний сценарій	Покупець переходить на сторінку корзини та натискає купити та чекає на підтвердження. Після підтвердження здійснюється оплата.

Таблиця 1.16 - Варіант використання UC016

Назва	Ведення аккаунту
Опис	Покупець корегує данні власного профілю.
Учасники	Покупець.
Передумови	Деякі данні були внесені некорректно або змінились.
Постумови	Дані користувча оновлені в системі.
Основний сценарій	Користувач переходить на власну сторінку, коригує дані та зберігає зміни.

Функціональні вимоги додатку описано наступними таблицями.

Таблиця 1.67 - Опис функціональної вимоги REQ001

Номер	REQ001
Назва	Ведення товарів.
Опис	Адміністратор може додавати видаляти та оновлювати товари.

Таблиця 1.78 - Опис функціональної вимоги REQ002

Номер	REQ002
Назва	Додавання покупців.
Опис	Адміністратор може додавати нових покупців у систему.

Таблиця 1.89 - Опис функціональної вимоги REQ003

Номер	REQ003
Назва	Здійснення купівлі.
Опис	Покупець може купувати товари.

Таблиця 1.20 - Опис функціональної вимоги REQ004

Номер	REQ004
Назва	Перегляд товарів з певної категорії.
Опис	Користувач може обрати категорію товарів і побачити всі товари з цієї категорії.

Таблиця 1.21 - Опис функціональної вимоги REQ005

Номер	REQ005
Назва	Сортування товарів за ціною.
Опис	Користувач може обрати порядок виведення товарів: від найдорожчих до найдешевших або навпаки.

Таблиця 1.22 - Опис функціональної вимоги REQ006

Номер	REQ006
Назва	Сортування товарів за новизною.
Опис	Користувач може обрати порядок виведення товарів: від тих щ обули додані або оновлені нещодавно до найстарших і навпаки.

Таблиця 1.93 - Опис функціональної вимоги REQ007

Номер	REQ007
Назва	Фільтрація товарів за ціною
Опис	Користувач може вказати діапазон цін для товарів і будуть виведені товари з цінами, що потрапляють у цей діапазон.

Таблиця 1.104 - Опис функціональної вимоги REQ008

Номер	REQ008
Назва	Пошук товарів.
Опис	Покупець може ввести назву товару у пошук та будуть виведені товари які у своїй назві містять запит користувача.

Таблиця 1.115 - Опис функціональної вимоги REQ009

Номер	REQ009
Назва	Коригування даних покупця.
Опис	Користувач може змінити власні дані.

Таблиця 1.126 - Опис функціональної вимоги REQ010

Номер	REQ010
Назва	Внесення параметрів покупця.
Опис	Покупець може внести власні параметриу систему.

Таблиця 1.137 - Опис функціональної вимоги REQ011

Номер	REQ011
Назва	Примірка товарів.
Опис	Покупець може приміряти товари.

Таблиця 1.148 - Опис функціональної вимоги REQ012

Номер	REQ012
Назва	Реєстрація
Опис	Гість може створити аккаунт покупця.

Таблиця 1.159 - Опис функціональної вимоги REQ013

Номер	REQ013
Назва	Вхід
Опис	Гість може увійти у існуючий покупця.

Взаємозв’язки між вимогами клієнтського застосунку відображені на рисунку 1.1.

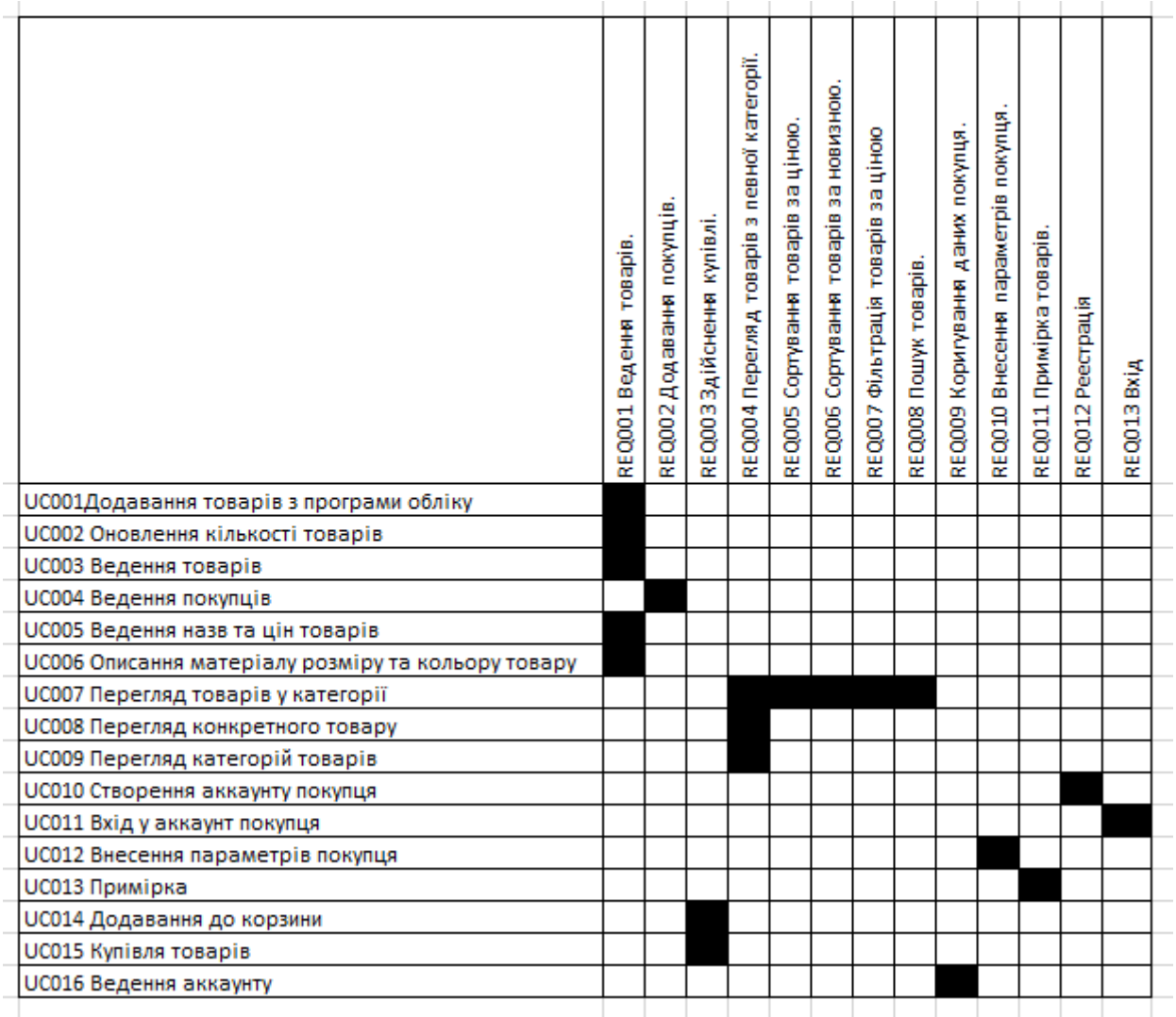


Рисунок 1.1 - Матриця залежності між вимогами застосунку і варіантами використання

1.4.2 Розроблення нефункціональних вимог

Вимоги до інтерфейсу:

– застосування має відображатись у браузері.

Апаратні та програмні вимоги:

– застосування повинно запускатися на пристроях з кольоровим дисплеєм (вимога для візуалізації);

– застосування має підтримуватись такими браузерами: Google Chrome версії 60 і вище, Internet Explorer версії 10 і вище, Firefox версії 4 і вище.

Операційні вимоги:

– продуктивність:

1) кількість запитів в сервісі на секунду не перевищує 100;

2) час відповіді системи на запит статистичного аналізу не більший ніж 1 секунда;

3) час відповіді системи на запит візуалізації не більший ніж 2 секунди.

1.4.3 Постановка комплексу завдань модулю

Розроблюване веб-застосування призначене для вирішення проблеми відсутності примірки у онлайн-магазині.

Мета створення даної роботи – створення такого веб-застосування для продажу одягу, в якому покупець зможе вказавши свої характеристики подивитись як на ньому виглядає модель одягу.

Для досягнення мети даної роботи система повинна вирішувати наступні задачі:

– створення сервісу що на основі характеристик одягу генерує стилізоване зображення одягу та надає можливість перефарбувати стилізоване зображення в інший колі;

– створення сервісу що на основі вимірів людини генерує стилізоване зображення людини з відповідними пропорціями та надягає на стилізоване зображення людини стилізоване зображення одягу;

– створення веб-застосування для продажу одягу з інтеграцією цих сервісів;

– інтеграція з існуючою базою даних.

Веб-застосування має підтримуватись такими браузерами: Google Chrome версії 60 і вище, Internet Explorer версії 10 і вище, Firefox версії 4 і вище.

Математичне забезпечення

Алгоритм генерації зображення людської фігури

Алгоритм генерує зображення людини за допомогою простих геометричних фігур таких як еліпс, гіпербола, парабоа, пряма, та ін. Для генерації жіночої фігури потрібні наступні параметри: обхват шиї (Ош), довжина плеча (Дп), обхват грудей (Огр), обхват під грудьми(Опг) – тільки для жінок, висота грудей(Вг) – тільки для жінок, центр грудей (Цг) – тільки для жінок, ширина спини(Шс), висота від шиї до талії (Вшт), обхват талії (От), обхват стегон (Ост), Обхват стегна (Ос), довжина стегна (Дс), обхват ноги (Он), обхват коліна (Ок), обхват гомілки (Ог), обхват щиколотки (Ощ), довжина ноги (Дн), обхват плеча (Оп), обхват зап’ястя (Оз), довжина рукова до ліктя (Дрл), довжина рукава (Др). Для чоловічої фігури обхвад під грудьми, центр грудей, та висота грудей не потрібні.

Опис алгоритму:

– генерація голови;

– генерація ший на основі Ош, вираховуються точки шиї;

– генерація плечей на основі Дп то точок шиї, вираховуються точки плечей;

– якщо генерується жіноче зображення, то генерація грудей на основі Огр, Опг Вг, Цг, Шс, обрахунок точок під грудьми, генерація тіла на основі Вшт, От, Ост, Ос, Дс та точок під грудьми, вираховуються точки стегон;

– якщо генерується чоловіче зображення, то генерація тіла на основі Огр, От, Вшт, Ост, Ос, Дс, вираховуються точки стегон;

– генерація ніг на основі Он, Ок, Ог, Ощ, Дн та точок стегон;

– генерація рук людини на основі Оп, Оз, Дрл, Др та точок плечей.

1.5.2 Алгоритм генерації зображення одягу на людині

Розроблено 2 алгоритми які генерують цілу одежу на верхню половину тіла – футболка, светр, та на нижню половину тіла – шорти, штани.

Для одежі на верхню половину тіла використовуються такі параметри: глибина вирізу (Гв), довжина вирізу (Дв), довжина плечей(Дп), довжина рукава (Др), обхват рукава (Ор), якщо одежа має довгі рукава, то другий обхват рукава (Ор2), обхват грудей (Ог), обхват нижнього краю вирбу (Он), обхват середини виробу(Ос), довжина виробу (Двр), матеріал. Матеріал – це коєфіцієнт ростягування тканини, якщо він рівний 1 – одяг не ростягується, наприклад льон, якщо 2 – одяг може розтягнутися у 2 рази, наприклад трикотаж.

Опис алгоритму для одежі на верхню половину тіла:

– згенерувати виріз виробу;

– вирахувати де опинятся плечі виробу та сгенерувати шов рукава на основі Дп;

– сгенерувати рукава на основі Др, Ор, якщо є то Ор2, плечей виробу та матеріалу;

– сгенерувати як сяде одяг на тілі на основі Ог, Ос, Он, Двр та матеріалу.

Для одежі на нижню половину виробу використовуються такі параметри: висота посадки (Вп), обхват верхнього краю виробу (Ов), довжина штанини (Дш), обхват одного з нижніх країв виробів (Он), якщо це бриджі або штани, то обхват на відстані 0.15 довжини виробу від нижнього краю (О14), якщо це штани, то обхвт на відстаі 0.25 від нижнього краю виробу (О34), матеріал.

Опис алгоритму:

- вирахувати де опинится верхній край виробу;
- вирахувати де опинится початок штанин;
- згенерувати посадку штанів на основі матеріалу;
- згенерувати штанини на основі параметрів і матеріалу.

1.6 Висновки до розділу

В розділі 1 проведено:

- аналіз існуючих технічних засобів і відомих рішень;
- аналіз відомих програмних продуктів, їх переваг та недоліків;
- аналіз вимог до програмного забезпечення – розроблені функціональні та нефункціональні вимоги.

2 МОДЕЛЮВАННЯ ТА КОНСТРУЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Моделювання та аналіз програмного забезпечення

Застосування було розроблено для Google Chrome версії 60 і вище, Internet Explorer версії 10 і вище, Firefox версії 4 і вище. Застосування було розроблено на мовах програмування java та python. На Java була реалізована взаємодія з базою даних та бізнес-логіка процесів ведення товарів та покупців. На python було розроблено модуль дла генерації зображення людини і одягу.

В застосуванні реалізована трьохшарова архітектура що дозволяє виокремити незалежні модулі та окремо розширювати кожен з них. Сервіс що відповідає за примірку працює окремо і основне застосування звертається до нього коли є відповідний запит.

Модель бізнес-процесів представлена на рисунку 2.1.

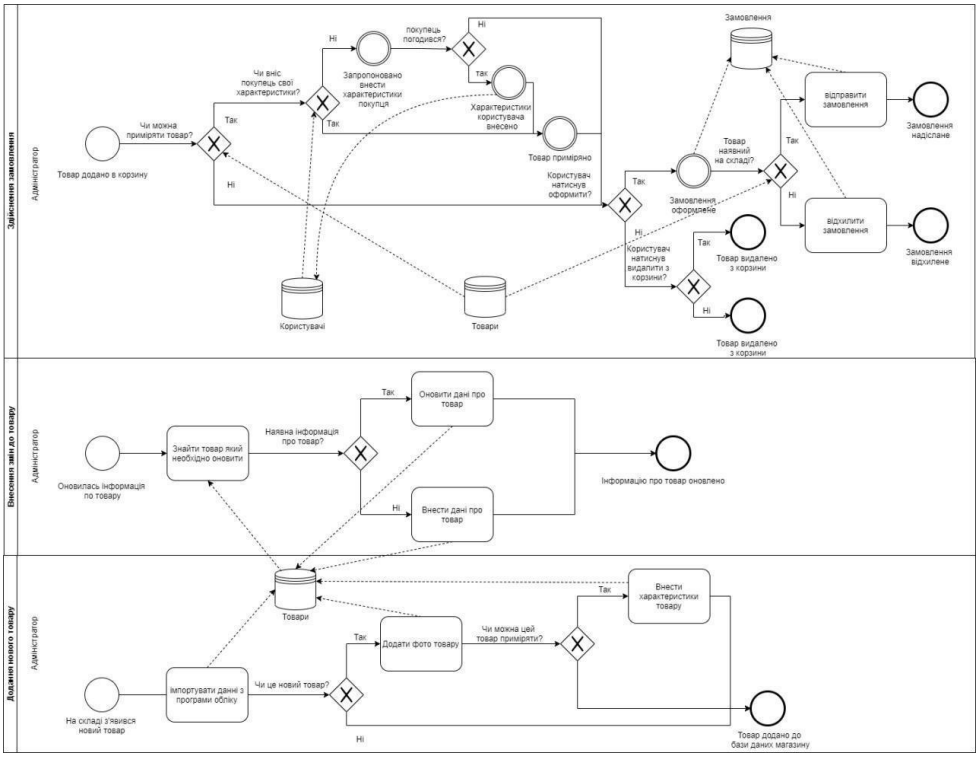


Рисунок 2.1 - Схема структурна бізнес-процесів

2.2 Архітектура програмного забезпечення

Діаграма класів програмного продукту наведена у графічному матеріалі. Детальний опис класів та функцій наведений нижче в таблицях.

Таблиця 2.1 - Опис класів застосування

Клас	Опис
AdminController	Клас-контролер для функції адміністратора таких як додавання чи оновлення продуктів.
AuthController	Клас-контролер для реєстрації та авторизації користувачів.
ShopController	Клас-контролер для показу товарів.
UserController	Клас-контролер для управління параметрами користувача.
CustomerService	Клас для створення користувача.
Клас	Опис
ProductService	Клас для отримання, створення, редагування та видалення товарів.
UserService	Клас для створення аккаунту.
JwtAuthenticationFilter	Клас для фільтрування запитів.
JwtTokenProvider	Клас для обробки токенів.
SecurityConfig	Клас для конфігурації налаштувань доступу до ресурсів.
UserDetailsServiceImp	Клас для отримання інформації про аккаунт.
CsvData	Клас що є представленням даних отриманих з програми обліку.
CsvProduct	Клас що є представленням продукту, отриманого з прграми обліку.
InvalidParamsException	Виключення що виникає при некоректно переданих параметрах.

Продовження таблиці 2.1

Клас	Опис
ShopExeption	Абстрактне виключення.
UserExsistException	Виключення що виникає при спробі зареєструвати нового користувача на існуючу пошту.
Customer	Клас що є представленням користувача в системі.
ProductModel	Клас що є представленням товарів у системі.
ProductName	Клас що є представленням окремих видів товару у системі.
User	Клас що є представленням акаунту користувача в системі.
DrowService	Клас що генерує зображення людини та зображення одягу.

Таблиця 2.2 - Опис методів класів сервісу

Клас	Метод	Опис
AdminController	importProducts	Імпортує данні про товари з csv файлу. Повертає статус операції імпорту. Параметри: - csvData – CsvData – данні з csv файлу.
AdminController	getAllProducts	Передає продукти з серверу на фронтенд. частину. Повертає статус операції передачі. Параметри: - page – int – номер сторінки з товарами; - size – int – кількість товарів на сторінці; - sort – String – сортування товарів.

Продовження таблиці 2.2

Клас	Метод	Опис
AdminController	editProduct	Редагування параметрів товару. Повертає статус операції редагування. Параметри: - product – ProductName – товар з новими параметрами.
AdminController	editProduct	Редагування параметрів конкретних єкземплярів товару, наприклад ціни на штани s розміру. Повертає статус операції редагування. Параметри: product – ProductName – товар з новими параметрами.
AuthController	login	Авторизує користувача у системі. Повертає статус авторизації. Параметри: - user – User – данні користувача для авторизації
AuthController	addCustomer	Створює нового користувача в системі. Повертає статус операції створення. Параметри: - customer – Customer – данні користувача для створення нового акаунту.

Продовження таблиці 2.2

Клас	Метод	Опис
ShopController	getProductsForShop	Передача продуктів за критеріями на фронтенд. Повертає результат операції передачі. Параметри: - name – String – назва товару; - minPrice – BigDecimal – мінімальна ціна товару; - maxPrice – BigDecimal – максимальна ціна товару; - color – String – колір товару; - page – int – кількість сторінок товару; - size – int – розмір сторінки.
UserController	getCurrentUser	Отримання даних поточного користувача. Повертає статус операції отримання.
Customer	getId	Отримання id користувача. Повертає id користувача.
Customer	setId	Встановлення id користувача. Параметри: - id – int – id користувача.
Customer	getFullName	Отримання повного ім’я користувача. Повертає повне ім’я користувача.
Customer	setFullName	Встановлення повного імені користувача. Параметри: - fullName – string – повне ім’я користувача.

Продовження таблиці 2.2

Клас	Метод	Опис
Customer	getPhone	Отримання мобільного номеру користувача. Повертає мобільний номер користувача.
Customer	setPhone	Встановлення мобільного номеру користувача. Парамети: - phone – string – мобільний номер користувача.
Customer	getUser	Отримання даних акаунту користувача. Повертає дані акаунту користувача.
Customer	setUser	Встановлення даних акаунту користувача. Параметри: - user – User – дані акаунту користувача.
ProductModel	c	Отримання id продукту. Повертає id продукту.
ProductModel	setId	Встановлення id продукту. Параметри: - id – int – id продукту.
ProductModel	getSize	Отримання розміру продукту. Повертає розмір продукту.
ProductModel	setSize	Встановлення розміру продукту. Параметри: - size – Size – розмір одягу.
ProductModel	getPrice	Отримання ціни продукту. Повертає ціну продукту.

Продовження таблиці 2.2

Клас	Метод	Опис
ProductModel	setPrice	Встановлення ціни продукту. Параметри: - price – BigDecimal – ціна товару.
ProductModel	getProduct Name	Отримання назви продукту. Повертає назву продукту.
ProductModel	setProductN ame	Встановлення назви продукту. Параметри: - productName – ProductName – назва товару.
ProductName	getId	Отримання id користувача. Повертає id користувача.
ProductName	setId	Встановлення id користувача. Параметри: - id – int – id користувача.
ProductName	getName	Отримання назви продукту. Повертає назву продукту.
ProductName	setName	Встановлення назви продукту. Параметри: - name – string – назва товару.
ProductName	getColor	Отримання кольору товару. Повертає колір товару.
ProductName	setColor	Встановлення кольору товару. Параметри: - color – string – колір товару.
ProductName	getProduct Models	Отримання всіх розмірів і цін товару. Повертає пари розмір-ціна товару.

Продовження таблиці 2.2

Клас	Метод	Опис
ProductName	setProduct Models	Встановлення всіх розмірів і цін товару. Параметри: - productModels – set<ProductModels> - пари розмір-ціна для цього товару.
User	getPasswor d	Отримання паролю. Повертає пароль користувача.
User	getUsernam e	Отримання пошти користувача. Повертає пошту користувача.
User	isAccountN onExpired	Перевірка, чи існує ще акаунт. Повертає true якщо він існує і false – якщо ні.
User	isAccountN onLocked	Перевірка, чи не заблоковано акаунт. Повертає true якщо він не заблокований і false – якщо заблокований.
User	isCredential sNonExpire d	Перевірка, чи не застарілі дані акаунту. Повертає true якщо вони не застарілі і false – якщо застарілі.
User	isEnabled	Перевірка, чи доступний акаунт. Повертає true якщо вон доступний і false – якщо ні.
User	getId	Отримання id акаунту. Повертає id акаунту.
User	setId	Встановлення id акаунту. Параметри: - id – int – id акаунту.
User	getEmail	Отримання пошти користувача. Повертає пошту користувача.
User	setEmail	Встановлення пошти користувача. Параметри: - email – string – пошта користувача.

Продовження таблиці 2.2

Клас	Метод	Опис
User	setPassword	Встановлення паролю користувача. Параметри: - password – string – пароль користувача.
User	getRole	Отримання ролі користувача. Повертає роль користувача.
User	setRole	Встановлення ролі користувача. Параметри: - role – string – роль користувача.
InvalidParamsException	InvalidParamsException	Конструктор. Параметри: - message – string – помилка.
ShopException	ShopException	Конструктор. Параметри: - message – string – помилка.
UserExistsException	UserExistsException	Конструктор. Параметри: - message – string – помилка.
CsvData	getCsvProducts	Отримання даних про товари. Повертає дані про товари.
CsvData	setCsvProducts	Встановлення даних про товари. Параметри: - csvProducts – List<CsvProduct> - список товарів.
CsvProduct	getCode	Отримання коду товару. Повертає код товару.

Продовження таблиці 2.2

Клас	Метод	Опис
CsvProduct	setCode	Встановлення коду товару. Параметри: - code – long – код товару.
CsvProduct	getName	Отримання назви товару. Повертає назву товару.
CsvProduct	setName	Встановлення назви товару. Параметри: - name – string – назва товару.
CsvProduct	getCost	Отримання собівартості товару. Повертає собівартість товару.
CsvProduct	setCost	Встановлення собівартості товару. Параметри: - cost – bigDecimal – собівартість товару.
CsvProduct	getMarkup	Отримання націнки товару. Повертає націнку товару.
CsvProduct	setMarkup	Встановлення націнки товару. Параметри: - markup – bigDecimal – націнка товару.
CsvProduct	getPrice	Отримання ціни на товар. Повертає ціну на товар.
CsvProduct	setPrice	Встановлення ціни товару. Параметри: - price – bigDecimal – ціна на товар.
CsvProduct	getAmount	Отримання кількості товару. Повертає кількість товару.

Продовження таблиці 2.2

Клас	Метод	Опис
CsvProduct	setAmount	Встановлення кількості товару. Параметри: - amount – bigDecimal – кількість товару.
CsvProduct	getAddDate	Отримання дати додання товару. Повертає дату додання товару.
CsvProduct	setAddDate	Встановлення дати додання товару. Параметри: - date – Data – дата додання товару.
CsvProduct	getColor	Отримання кольору товару. Повертає колір товару.
CsvProduct	setColor	Встановлення кольору товару. Параметри: - color – string – колір товару.
CsvProduct	getSize	Отримання розміру товару. Повертає розмір товару.
CsvProduct	setSize	Встановлення розміру товару. Параметри: - size – string – розмір товару.
CsvProduct	toProduct	Перетводить продукт у форматі з програми обліку у продукт формату веб-застосування. Повертає продукт.

Продовження таблиці 2.2

Клас	Метод	Опис
JwtAuthenticationFilter	doFilterInternal	Фільтрація запитів. Параметри: - httpServletRequest – HttpServletRequest – запит; - httpServletResponse – HttpServletResponse – відповідь.
JwtAuthenticationFilter	getJwtFromRequest	Отримання токену с запиту. Повертає токен. Параметри: - httpServletRequest – HttpServletRequest – запит;
JwtTokenProvider	createToken	Створення токену. Повертає токен. Параметри: - authentication – Authentication – данні акаунту.
JwtTokenProvider	getUserIdFromToken	Отримання id акаунту за токеном. Повертае id користувача. Параметри: - tocken – string – токен.
JwtTokenProvider	validateToken	Перевіряє, чи валідний токен. Повертає true якщо токен валідний та false – якщо ні. Параметри: - tocken – string – токен.
UserDetailServiceImpl	loadUserByName	Отримання користувача за поштою. Повертає користувача. Параметри: - name – string – пошта.

Продовження таблиці 2.2

Клас	Метод	Опис
UserDetailServiceImpl	loadUserById	Отримання користувача за id. Повертає користувача. Параметри: - id -int – id.
CustomerService	AddCustomer	Додає користувача у базу даних. Параметри: - customer – Customer – данні користувача.
ProductService	importProducts	Записує данні про товари у базу даних. Параметри: - csvData – CsvData – данні з програми обліку.
ProductService	getAllProducts	Отримує всі товари з бази даних. Повертає список товарів.
ProductService	getAllProductsCount	Отримує загальну кількість товарів. Повертає загальну кількість товарів.
ProductService	editProductName	Редагування товару. Параметри: - product – ProductName – товар з новими параметрами.
ProductService	editProductModel	Редагування цін на конкретні розміри товару. Параметри: - product – ProductModel - товар з новими параметрами.
UserService	getCurrentUser	Отримує данні про аутентифікованого користувача. Повертає данні про користувача.

Продовження таблиці 2.2

Клас	Метод	Опис
Figures	FromSmToPixel	Конвертує значення довжини у сантиметрах у значення довжини у пікселях. Повертає значення довжини у пікселях. Параметри: - value – double – довжина у сантиметрах.
Figures	FromPixelToSm	Конвертує значення довжини у пікселях у значення довжини у сантиметрах. Повертає значення довжини у пікселях. Параметри: - value – int – довжина у пікселях.
Figures	DrowCircle	Малює фрагмент кола із зазначеним радіусом у межах заданих кутів. Параметри: - img – numpy.array – зображення; - R – double – радіус кола; - X – double – координата центру кола по осі x. - Y – double – координата центру кола по осі y. - start – double – кут початку фрагмента кола; - end – double – кут кінця фрагмента кола.

Продовження таблиці 2.2

Клас	Метод	Опис
Figures	DrowParabola	Малює фрагмент параболи за заданими точками і межами. Параметри: - img – numpy.array – зображення; - points – numpy.array – точки через які має проходити парабола; - borders - numpy.array – межі у яких малювати параболу.
Figures	DrowHiperbolaByPoints	Малює фрагмент гіперболи яка проходить через задані точки. Параметри: - img – numpy.array – зображення; - width – відстань між гілками гіперболи; - X – double – координата точки через яку має проходити гіпербола по осі x; - Y – double – координата точки через яку має проходити гіпербола по осі y; - point – numpy.array – точка через яку має проходити гіпербола; - YB – double – початок проміжку для зображення гіперболи; - YE – double – кінець проміжку для зображення гіперболи.

Продовження таблиці 2.2

Клас	Метод	Опис
Figures	DrowHiperbolaByBParam	Малює фрагмент гіперболи яка проходить через задані точки та має заданий параметр. Параметри: - img – numpy.array – зображення; - width – відстань між гілками гіперболи; - X – double – координата точки через яку має проходити гіпербола по осі x; - Y – double – координата точки через яку має проходити гіпербола по осі y; - b2 – double – параметр гіперболи; - YB – double – початок проміжку для зображення гіперболи; - YE – double – кінець проміжку для зображення гіперболи.
BodyBuilder	GetHiperbolaPointByBParam	Обчислення координат x які ввідповідають заданій гіперболі та координаті y. Параметри: - img – numpy.array – зображення; - width – відстань між гілками гіперболи; - X – double – координата точки через яку має проходити гіпербола по осі x; - Y – double – координата точки через яку має проходити гіпербола по осі y; - b2 – double – параметр гіперболи; - Y1 – double – координата y шуканих точок;

Продовження таблиці 2.2

Клас	Метод	Опис
BodyBuilder	DrowBrunch	Малює гілку параболи за заданими точками. Параметри: - img – numpy.array – зображення; - points – numpy.array – точки через які має проходити гілка параболи.
BodyBuilder	DrowHead	Малює голову людини. Параметри: - img – numpy.array – зображення.
BodyBuilder	DrowNeck	Малює шию людини. Повертає структурні точки зображення. Параметри: - img – numpy.array – зображення; - circumference – double – обхват шиї людини.
BodyBuilder	DrowShoulders	Малює плечі людини. Повертає структурні точки зображення. Параметри: - img – numpy.array – зображення; - length – довжина плеча людини; - neck – параметри шиї людини;

Таблиця 2.3 - Опис таблиць бази даних

Таблиця	Опис
Users	У таблиці зберігається інформація необхідна для авторизації користувача.
Customers	У таблиці зберігається особиста інформація користувача.

Продовження таблиці 2.3

Таблиця	Опис
Orders	У тоблиці зберігаються замовлення користувачів.
Deal	Службова таблиця яка представляє собою купівю одного товару з усього замовлення.
Products	У таблиці зберігається конкретні співвідношення цін та розмірів товарів.
Product_names	У таблиці зберігається опис товарів.
Structures	У таблиці зберігається склад товарів.

Таблиця 2.4 – Опис полів бази даних

Таблиця	Поле	Опис
Users	id	Первинний ключ таблиці Users. Тип: integer.
Users	email	Пошта користувача. Тип: character varying.
Users	password	Пароль користувача. Тип: character varying.
Users	role	Роль користувача: USER або ADMIN. Тип: character varying.
Customers	id	Первинний ключ таблиці Customers. Тип: integer.
Customers	user_id	Зовнішній ключ таблиці Customers на таблицю Users. Тип: integer.
Customers	full_name	Повне ім’я користувача. Тип: character varying.

Продовження таблиці 2.4

Таблиця	Поле	Опис
Customers	phone	Мобільний номер користувача. Тип: character varying.
Customers	discount	Персональна знижка користувача. Тип: character varying.
Customers	params	Параметри користувача. Тип: json.
Orders	id	Первинний ключ таблиці Orders. Тип: integer.
Orders	customer_id	Зовнішній ключ таблиці Orders на таблицю Customers. Тип: integer.
Orders	date	Дата оформлення замовлення. Тип: date.
Orders	status	Стан замовлення. Тип: character varying.
Deal	id	Первинний ключ таблиці Deal. Тип: integer.
Deal	order_id	Зовнішній ключ таблиці Deal на таблицю Orders. Тип: integer.
Deal	product_id	Зовнішній ключ таблиці Deal на таблицю Products. Тип: integer.
Deal	price	Ціна товару. Тип: money.
Products	id	Первинний ключ таблиці Products. Тип: integer.

Продовження таблиці 2.4

Таблиця	Поле	Опис
Products	product_name_id	Зовнішній ключ таблиці Products на таблицю Product_names. Тип: integer.
Products	size	Розмір товару : XS, S, M, L, XL. Тип: character varying.
Products	price	Ціна товару. Тип: money.
Products	params	Параметри одягу. Тип: json.
Product_names	id	Первинний ключ таблиці Product_names. Тип: integer.
Product_names	category	Категорія, до якої належить товар. Тип: character varying.
Product_names	name	Назва товару. Тип: character varying.
Product_names	color	Колір товару. Тип: character varying.
Product_names	photo	Фото товару. Тип: character varying.
Product_names	status	Чи є товар у продажу. Тип: boolean.
Structures	id	Первинний ключ таблиці Structures. Тип: integer.
Structures	product_name_id	Зовнішній ключ таблиці Structures на таблицю Product_names. Тип: integer.

Продовження таблиці 2.4

Таблиця	Поле	Опис
Structures	material	Назва матеріалу. Тип: character varying.
Structures	percent	Частка матеріалу у тканині. Тип: integer.

Аналіз безпеки даних

Безпека даних користувача – це дуже важливе питання, коли мова йде про інтернет-магазин. Для безпеки даних користувача на рівні бази даних модель користувача було розділено на дві частини: дані необхідні для авторизаціїї та власне особисті дані користувача. Паролі у базі даних зберігаються у хешованому та зашифрованому вигляді. Це дозволить встигнути безпечно замінити паролі користувачів якщо виникне така ситуаія, що таблицю паролів було вражено.

Для безпечного з’єднання використовується протокол https[7], що на відміну від звичайного http використовує ssh з’єднання[8] що шифрується і є більш безпечним.

Висновки до розділу

У розділі було розглянуто:

- базу даних застосування;
- архітектуру застосування;
- фукціональність застосування;
- безпеку застосування.

3 АНАЛІЗ ЯКОСТІ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Вступ

Оцінка якості програмного забезпечення є дуже важливою частиною процесу розробки. Аналіз якості програмного забезпечення здійснюється за допомогою оцінки того, на скільки воно відповідає вимогам. Оцінка здійснюється шляхом тестування функціональних і нефункціональних вимог.

Функціональність, що підлягає тестуванняю

- перегляд товарів за категоріями;
- сортування товарів за ціною;
- сортування товарів за новизною;
- пошук товарів за назвою;
- фільтрація товарів за ціною;
- примірка товарів;
- додання товарів у корзину.

Методологія проведення тестування

Тестування має покрити всю функціональність що підлягає тестуванню. Для цього слід протестувати всі варіанти використання застосування, зазначені у діаграмі використання. Для кожного тесту мають бути записані передумови, постумови, очікуваний та отриманий результат. Звіти з тестів мають бути записані у GitLab[9]. Звіти мають бути передані розробнику для усунення виявлених помилок.

Критерії проходження/провалення тестування

Покриття тестами має складати більше ніж 90% проекту. Основні варінти використання застосування мають працювати як очікуволося, тобто як

зазначено у вимогах. Більше 80% тестів повинні бути пройденими, а виявлені помилки не повинні бути критичними для використання веб-застосування.

3.5 Процес тестування

Протестованими мають буті тесткейси, наведені у таблицях 3.1 – 3.10

Таблиця 3.1 – ТС001

Назва	Експорт товарів
Передумови	З програми обліку експортовано файл у csv форматі, адміністратор зайшов на сторінку адміністратора
Дія	Адміністратор натискає єкспортувати, обирає файл у csv форматі з програми обліку.
Очікуваний результат	Данні з файлу внесені у базу даних.

Таблиця 3.2 – ТС002

Назва	Редагування товарів
Передумови	Товар існує у базі даних, адміністратор зайшов на сторінку адміністратора.
Дія	Адміністратор натискає редагувати, вносить нові дані та натискає оновити.
Очікуваний результат	Дані про товар оновлені.

Таблиця 3.3 – ТС003

Назва	Зміна статусу замовлення
Передумови	Покупець оформив замовлення, адміністратор зайшов на сторінку адміністратора, товар є на складі.

Продовження таблиці 3.3

Дія	Адміністратор змінює статус товару на відпралено.
Очікуваний результат	Статус замовлення змінено на відправлено.

Таблиця 3.4 – ТС004

Назва	Створення аккаунту
Передумови	Користувач ззайшов у систему як гість
Дія	Користувач натиснув реестрація, вніс пошту та пароль та натиснув зареєструватись.
Очікуваний результат	Створено новий аккаунт користувача.

Таблиця 3.5 – ТС005

Назва	Вхід у аккаунт
Передумови	Користувач ззайшов у систему як гість.
Дія	Користувач натиснув вхід, вніс пошту та пароль та натиснув ввійти.
Очікуваний результат	Користувача авторизовано.

Таблиця 3.6 – ТС006

Назва	Перегляд товарів
Передумови	Користувач ззайшов у систему як гість або авторизувався, у системі наявні товари.

Продовження таблиці 3.6

Дія	Користувач натискає на категорію товарів, які хоче переглянути.
Очікуваний результат	Виведено товари у категорії, яку натиснув користувач.

Таблиця 3.7 – ТС007

Назва	Перегляд конкретного товару
Передумови	Користувач ззайшов у систему як гість або авторизувався, у системі є товар.
Дія	Користувач натиснув на товар.
Очікуваний результат	Відображено інформацію про товар.

Таблиця 3.8 – ТС008

Назва	Додавання товару у корзину
Передумови	У системі є товар, користувач авторизувався у системі та перейшов на сторінку товару.
Дія	Користувач натискає додати в корзину.
Очікуваний результат	Товар додано в корзину.

Таблиця 3.9 – ТС009

Назва	Примірка товару
Передумови	Користувач авторизувався у системі, у користувача в корзині є товар, що можна приміряти.
Дія	Користувач натискає приміряти.
Очікуваний результат	Товар приміряно.

Таблиця 3.10 – ТС010

Назва	Оформлення замовлення
Передумови	Користувач авторизувався у системі, у корзині є товари.
Дія	Користувач натиснув оформити замовлення.
Очікуваний результат	Замовлення оформлене.

3.6 Результати тестування

Тестування було проведене за сценаріями наведеними у таблицях 3.1 – 3.10. Результати тестування наведені у таблиці 3.11.

Таблиця 3.11

Назва тесту	Очікуваний результат	Реальний результат	Висновок
Експорт товарів	Данні з файлу внесені у базу даних.	Данні з файлу внесені у базу даних.	Успішно.
Редагування товарів	Дані про товар оновлені.	Дані про товар оновлені.	Успішно.
Зміна статусу замовлення	Статус замовлення змінено на відправлено.	Статус замовлення змінено на відправлено.	Успішно.
Створення аккаунту	Створено новий аккаунт користувача.	Створено новий аккаунт користувача.	Успішно.
Вхід у аккаунт	Користувача авторизовано.	Користувача авторизовано.	Успішно.
Перегляд товарів	Виведено товари у категорії, яку натиснув користувач.	Виведено товари у категорії, яку натиснув користувач.	Успішно.

Продовження таблиці 3.11

Назва тесту	Очікуваний результат	Реальний результат	Висновок
Перегляд конкретного товару	Відображено інформацію про товар.	Відображено інформацію про товар.	Успішно.
Додавання товару у корзину	Товар додано в корзину.	Товар додано в корзину.	Успішно.
Примірка товару	Товар приміряно.	Товар приміряно.	Успішно.
Оформлення замовлення	Замовлення оформлене.	Замовлення оформлене.	Успішно.

3.7 Висновки до розділу

У даному розділі було розглянуто вимоги які необхідно протестувати методологію тестування та критерії оцінки результатів тестування. Також програмне забезпечення було протестовано за наведеною методологією.

4 ВПРОВАДЖЕННЯ ТА СУПРОВІД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Розгортання програмного забезпечення

Розроблене програмне забезпечення інтегроване з програмним забезпеченням «АРМ Торгівля V 2.0.1»[10], яке являє собою програму обліку товарів та містить модулі управління складом та касою. Інтеграція реалізована за рахунок експортування файлів з «АРМ Торгівля V 2.0.1» у інтернет-магазин.

Послідовність розгортання є наступною:

- розгорнути базу даних postgres на віддаленому сервері;
- розгорнути інтернет-магазин на віддаленому сервері, додаткового створення бази даних та таблиць не потрібно, оскільки прописані міграції;
- внести данні з програми обліку у інтернет-магазин;
- внести додаткові характеристики товарів, такі як їх фото та параметри необхідні для примірки у інтернет-магазин.

4.2 Робота з програмним забезпеченням

Оскільки інтеграція з програмою обліку реалізована через експортування файлів, при додванні або оновленні товарів адміністратор повинен експортувати ці данні у інтернет-магазин. Також адміністратор повинен періодичну уточнювати реальну кількість товарів, оскільки вони продаються як у звичайному, так і у інтернет-магазині.

Коли покупець оформлює замовлення, воно з’являється у переліку замовлень на сторінці адміністратора. Адміністратор має передзвонити покупцю та підтвердити замовлення, або відхилити замовлення, якщо товар відсутній на складі. Адміністратор може знайти замовлення за номером замовлення.

У інтернет-магазині можна додатково корегувати параметри товарів, такі як ціну на конкретні розміри, фото та параметри для примірки.

4.3 Висновки до розділу

В розділі 4 описано необхідні умови для розгортання розробленого програмного забезпечення та роботи з ним.

ВИСНОВКИ

Отже інтернет-магазин – це доволі розповсюджений спосіб продажу товарів, у тому числі і одягу. Онлайн формат дозволяє економити на приміщенні та не обмежувати свою аудиторію власним місцем розміщення. Проте виникає проблема того, що люди купують одяг не знаючи, як він на них сидить, і це призводить до того, що товари часто повертають. Для вирішення цієї проблеми у даній роботі розроблюється онлайн-примірочна.

Для створення більш точної моделі того як одяг сидітиме на людині, людина має внести доволі багато своїх вимірів, а продавець – багато вимірів одягу. Оскільки це застосування розроблялось для ательє, і одяг там не купують а шиють за викройками, то ці данні вже наявні у продавця і їх необхідно спеціально вимірювати. Онлайн-примірочна спочатку створює модель покупця, а потім надягає на неї вказаний одяг.

Онлайн-примірочна інтегрована у інтернет-магазин, проте є незалежним модулем, що дозволяє як використовувати її для різних магазинів, так і окремо модифікувати.

Такий сервіс дозволить виокремити інтернет-магазин серед інших та допоможе змогу покупцю зрозуміти, чи підходить йому те що він обрав, чи воно сидить не так, як він очікував.

ПЕРЕЛІК ПОСИЛАНЬ

1) Матеріали четвертої всеукраїнської науково-практичної конференції молодих вчених та студентів "інформаційні системи та технології управління". – 2020. – С. 137–138.

2) Розміри одягу, співвідношення розмірів [Електронний ресурс] – Режим доступу до ресурсу: http://club58.com.ua/pol6_info_2.htm.

3) Огляд технології від Amazon [Електронний ресурс] – Режим доступу до ресурсу: https://habr.com/ru/company/pochtoy/blog/412023/.

4) Приклад примірочної без урахування вимірів людини [Електронний ресурс] – Режим доступу до ресурсу: http://www.styleclub.com.ua/model_wardrobe.aspx#modeltop.

5) Приклад примірочної з реальними фото одягу [Електронний ресурс] – Режим доступу до ресурсу: https://showroom.onvolga.com/demo/fitting-clothes/.

6) PostgreSql [Електронний ресурс] – Режим доступу до ресурсу: https://www.postgresql.org/.

7) HTTPS [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/HTTPS.

8) SSH [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/SSH.

9) GitLab [Електронний ресурс] – Режим доступу до ресурсу: https://about.gitlab.com/.

10) АРМ Торгівля V 2.0.1 [Електронний ресурс] – Режим доступу до ресурсу: https://arm20.com/helpbase.php.

Факультет інформатики та обчислювальної техніки

Кафедра автоматизованих систем обробки інформації і управління

“ЗАТВЕРДЖЕНО”

В.о. завідувача кафедри

_____О.А. Павлов

“___”_____2020 р.

ВЕБ-ЗАСТОСУВАННЯ «ВІРТУАЛЬНЕ АТЕЛЬЄ»

Технічне завдання

КПІ.ІП-6314.045440-02-91

“ПОГОДЖЕНО”

Керівник проекту:

_____О.А Халус

Нормоконтроль:

_____К.І. Ліщук

Виконавець:

_____Л.В. Корольова

Київ – 2020 року

ЗМІСТ

1 НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ

Назва розробки: Веб-застосування «Віртуальне ательє»

Галузь застосування: інтернет-магазини одягу.

Наведене технічне завдання поширюється на розробку веб-застосування «Віртуальне ательє» КПІ.ІП-6315.045440.03.91, котра використовується для надання можливості інтернет-магазину продемонструвати як буде виглядати одяг на користувачі.

2 ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки веб-застосування «Віртуальне ательє» є завдання на дипломне проектування, затверджене кафедрою автоматизованих систем обробки інформації та управління Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського» (НТУУ «КПІ ім. Сікорського»).

3 ПРИЗНАЧЕННЯ РОЗРОБКИ

Розробка призначена для моделювання того як буде виглядати одяг на людині.

Метою розробки є створення web-додатку для продажу одягу, в якому буде можливість вказавши характеристики людини змоделювати те як на ній сяде одяг.

4 ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Вимоги до функціональних характеристик

Програмне забезпечення повинно забезпечувати виконання наступних основних функції:

Для користувача:

- здійснювати покупку товарів;
- здійснювати примірку деяких товарів;
- переглядати товари наявні у магазині.

Для адміністратора:

- вносити данні з програми обліку товарів;
- управляти продуктами.

Розробка має підтримуватись браузерами Google Chrome версії 60 і вище, Internet Explorer версії 10 і вище, Firefox версії 4 і вище.

Додаткові вимоги:

- для моделей одягу які можна приміряти мають бути виміряні їх параметри.

Вимоги до надійності

Передбачити контроль введення інформації.

Передбачити захист від некоректних дій користувача.

Забезпечити цілісність інформації в базі даних.

Умови експлуатації

Умови експлуатації згідно СанПін 2.2.2.542 – 96. Не висуваються.

Обслуговування Не висуваються.

Обслуговуючий персонал

Не висуваються.

Вимоги до складу і параметрів технічних засобів

Програмне забезпечення повинно функціонувати на серверах та комп'ютерах з операційною системою Windows/Ubuntu/MacOS.

Мінімальна конфігурація технічних засобів:

Тип процесору ..... Pentium.

Об'єм ОЗП 512 Мб.

Об'єм вільного дискового простору.....512 Мб.

Вимоги до інформаційної та програмної сумісності

Програмне забезпечення повинно працювати під управлінням операційних систем Windows 10/Ubuntu/MacOS.

Вхідні дані повинні бути представлені в наступному форматі: користувач вручну вводить данні у відповідні форми а також шлях до файлу у csv-форматі.

Результати повинні бути представлені в наступному форматі: сторінки веб-застосування.

Вимоги до маркування та пакування

Вимоги до маркування та пакування не пред'являються.

Вимоги до транспортування та зберігання

Вимоги до транспортування та зберігання не пред'являються.

Спеціальні вимоги

Згенерувати установчу версію програмного забезпечення.

5 ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ

Програмні модулі, котрі розробляються, повинні бути задокументовані, тобто тексти програм повинні містити всі необхідні коментарі.

Програмне забезпечення повинно мати довідникову систему

У склад супроводжувальної документації повинні входити наступні документи:

Пояснювальна записка не менше ніж на 100 аркушах формату А4 (без додатків 5.3.2 - 5.3.6).

Технічне завдання.

Керівництво користувача.

Програма та методика тестування.

Графічна частина повинна бути виконана у формату А3, котрі включаються у якості додатків до пояснювальної записки:

Схема структура варіантів використання.

Схема структурна бізнес-процесів.

Схема бази даних.

Схема структурна класів програмного забезпечення.

Креслення вигляду екранних форм.

6 СТАДІЇ І ЕТАПИ РОЗРОБКИ

Стадії та єтапи розробки наведені у таблиці 6.1.

Таблиця 6.1 - Стадії та єтапи розробки

№	Назва етапу	Строк	Звітність
1	Вивчення рекомендованої літератури	19.03.2020	
2	Аналіз існуючих методів розв’язання задачі	26.03.2020	
3	Постановка та формалізація задачі	26.03.2020	Технічне завдання
4	Аналіз вимог до програмного забезпечення	02.04.2020	Схема структурна програмного забезпечення та специфікація компонентів (діаграма класів, схема алгоритму)
5	Алгоритмізація задачі	02.04.2020	
6	Моделювання програмного забезпечення	09.04.2020	
7	Обґрунтування використовуваних технічних засобів	16.04.2020	
8	Розробка архітектури програмного забезпечення	23.04.2020	Схема структурна класів програмного забезпечення
9	Розробка програмного забезпечення	30.04.2020	Тексти програмного забезпечення
10	Налагодження програми	07.05.2020	Програма та методика тестування

Продовження таблиці 6.1

11	Виконання графічних документів	14.05.2020	Графічний матеріал проекту
12	Оформлення пояснювальної записки	21.05.2020	Пояснювальна записка проекту
13	Подання ДП на попередній захист	28.05.2020	
14	Подання ДП рецензенту	03.05.2020	
15	Подання ДП на основний захист	08.06.2020	

7 ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ

7.1. Види випробувань

Тестування розробленого програмного продукту виконується відповідно до “Програми та методики тестування”.

Факультет інформатики та обчислювальної техніки

Кафедра автоматизованих систем обробки інформації і управління

“ЗАТВЕРДЖЕНО”

В.о. завідувача кафедри

_____Олександр ПАВЛОВ

“___”_____2020 р.

ВЕБ-ЗАСТОСУВАННЯ «ВІРТУАЛЬНЕ АТЕЛЬЄ»

Програма та методика тестування

КПІ.ІП-6315. 045440.04.51

“ПОГОДЖЕНО”

Керівник проєкту:

_____О.А. Халус

Нормоконтроль:

_____К.І. Ліщук

Виконавець:

_____Л.В. Корольова

Київ – 2020 року

ЗМІСТ

1 ОБ’ЄКТ ВИПРОБУВАНЬ

Веб-застосування «Віртуальне ательє», яке являє собою веб-сайт, створений з використанням мови програмування java для власне веб-застосування та мови програмування python для сервісу примірки.

2 МЕТА ТЕСТУВАННЯ

У процесі тестування має бути перевірено наступне:

- функціональна працездатність елементів сторінок web-ресурсу;
- наявність доступу до бази товарів;
- забезпечення належного рівня безпеки даних;
- можливість здійснювати примірку;
- відповідність дизайну вимогам Технічного завдання.

3 МЕТОДИ ТЕСТУВАННЯ

Тестування виконується методом Gray Box Testing. Перевіряється як код, так і безпосередньо програмний продукт на відповідність функціональним вимогам. Тестування відбувається на рівні «системного тестування».

Використовуються наступні методи:

– функціональне тестування, зокрема на рівні Critical path test (ба-зове тестування);

– тестування продуктивності програмного забезпечення, зокрема Stability testing (тестування стабільності) та Load testing (навантажувальне тестування);

– тестування інтерфейсу.

4 ЗАСОБИ ТА ПОРЯДОК ТЕСТУВАННЯ

Тестування виконується засобами інструментарію Postman.

Працездатність web-ресурсу перевіряється шляхом:

– динамічного ручного тестування – введенням граничних та недопустимих значень в поля, які можна редагувати;

– динамічного ручного тестування на відповідність функціональним вимогам;

– статичного тестування коду;

– тестування web-ресурсу в різних web-браузерах;

– тестування при максимальному навантаженні;

– тестування стабільності роботи при різних умовах;

– тестування зручності використання;

– тестування інтерфейсу.

Факультет інформатики та обчислювальної техніки

Кафедра автоматизованих систем обробки інформації і управління

“ЗАТВЕРДЖЕНО”

В.о. завідувача кафедри

_____Олександр ПАВЛОВ

“___”_____2020 р.

«ВЕБ-ЗАСТОСУВАННЯ «ВІРТУАЛЬНЕ АТЕЛЬЄ»»

Керівництво користувача

КПІ.ІП-6315.045440.05.34

“ПОГОДЖЕНО”

Керівник проєкту:

_____О.А. Халус

Нормоконтроль:

_____К.І. Ліщук

Виконавець:

_____Л.В. Корольова

Київ – 2020 року

ЗМІСТ

1 ІНСТРУКЦІЯ КОРИСТУВАЧА

1.1 Зняття мірок

Для моделювання тіла користувача, він повинен зняти наступні мірки.

Обхват шиї: мірна стрічка проходить як на рисунку 1.1.

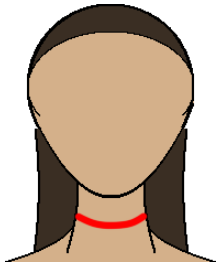


Рисунок 1.1 - Обхват шиї

Довжина плеча: вимірюють по лінії плеча від основи шиї до крайньої точки плеча (рисунок 1.2).

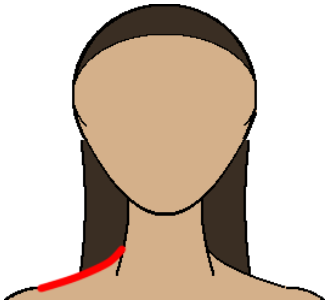


Рисунок 1.2 - Довжина плеча

Обхват грудей: мірна стрічка повинна проходити по виступаючих частин лопаток на спині і по найвищій частині грудей (рисунок 1.3).

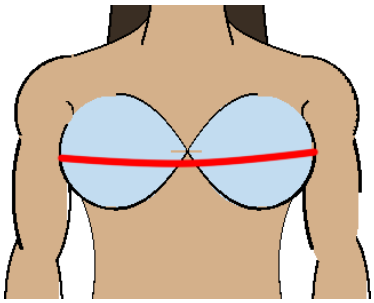


Рисунок 1.3 - Обхват грудей

Обхват під грудьми: мірна стрічка повинна проходити одразу під грудьми (рисунок 1.4). Мірка знімається тільки для жінок.

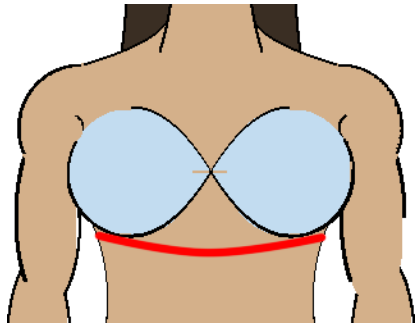


Рисунок 1.4 - Обхват під грудьми

Висота грудей: ця мірка вимірюється одночасно з міркою довжини переду до талії від лінії плеча у підставі шиї до виступаючої точки грудей (рисунок 1.5). Мірка знімається тільки для жінок.

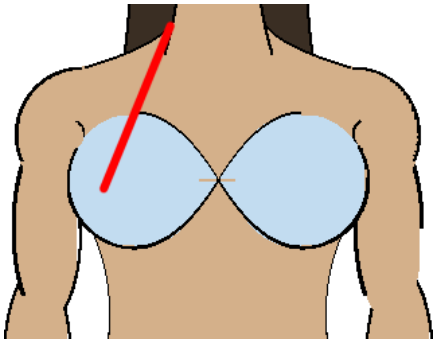


Рисунок 1.5 - Висота грудей

Центр грудей: вимірюють по горизонтальній лінії між виступаючими точками грудей (рисунок 1.6). Мірка знімається тільки для жінок.

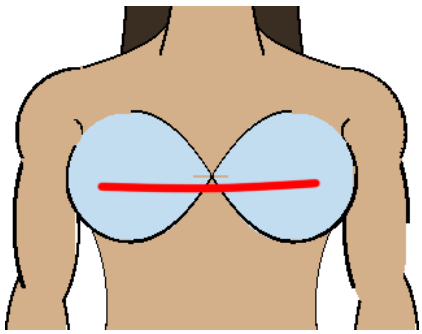


Рисунок 1.6 - Центр грудей

Обхват талії: вимірюють по самому вузькому місці талії (рисунок 1.7).

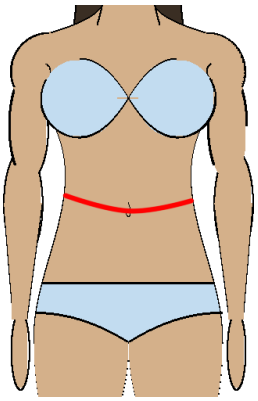


Рисунок 1.7 - Обхват талії

Довжина переду до талії: вимірюють від лінії плеча біля основи шиї через виступаючу точку грудей до лінії талії (рисунок 1.8)

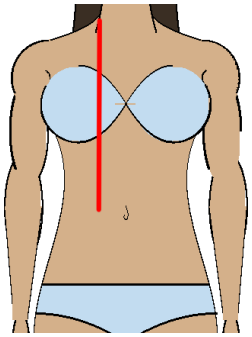


Рисунок 1.8 - Довжина переду до талії

Довжина стегна: вимірюють від самого вузького місця на талії до лінії по якій вимірюють обхват стегон (рисунок 1.9).

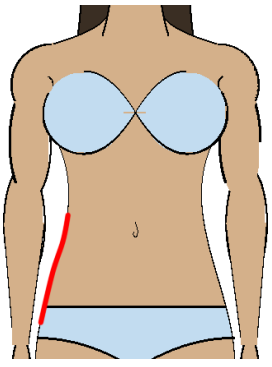


Рисунок 1.9 - Довжина стегна

Обхват стегон: вимірюють горизонтально по найбільш виступаючим точкам сідниць (рисунок 1.10).

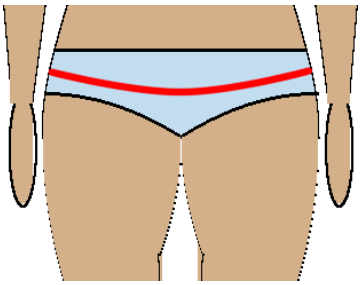


Рисунок 1.10 - Обхват стегон

Обхват стегна: вимірюють горизонтально у верхній частині ноги найширшу її частину (рисунок 1.11).

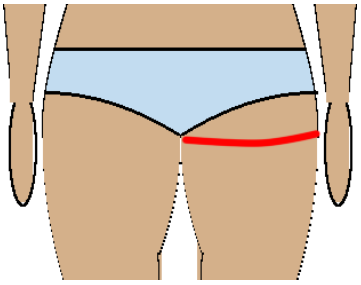


Рисунок 1.11 - Обхват стегна

Обхват ноги: вимірюють горизонтально окружність ноги вище коліна на 10-15 см (рисунок 1.12).

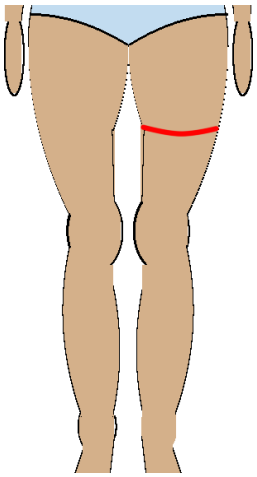


Рисунок 1.12 - Обхват ноги

Обхват коліна: вимірюють окружність коліна при зігнутою нозі під кутом 90°. Сантиметрова стрічка проходить по підколінної ямці, замикаючись спереду на середині коліна (рисунок 1.13).

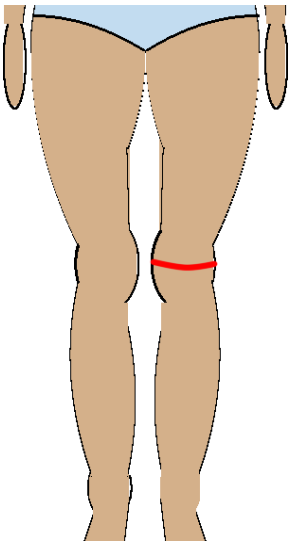


Рисунок 1.13 - Обхват коліна

Обхват ікри: вимірюють горизонтально по самим виступаючим точкам литкового м'яза (рисунок 1.14).

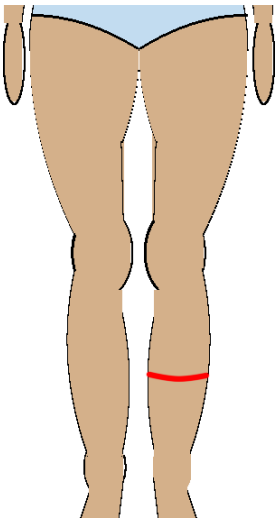


Рисунок 1.14 - Обхват гомілки

Довжина ноги: вимірювання проводять по внутрішньої сторони ноги від паху до підлоги при злегка розставлених ногах, прикладаючи лінійку ребром до паху. Вимірюють відстань від верхнього ребра лінійки вертикально до підлоги (рисунок 1.15)

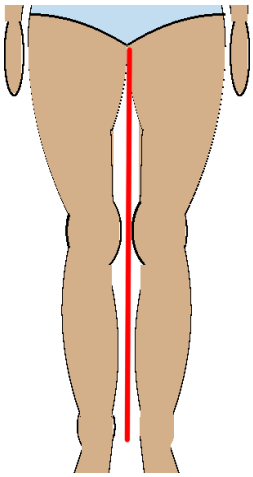


Рисунок 1.15 - Довжина ноги

Обхват щиколотки: вимірюють горизонтально навколо ноги над внутрішньою стороною щиколотки, мірна стрічка замикається на зовнішній поверхні гомілки (рисунок 1.16).

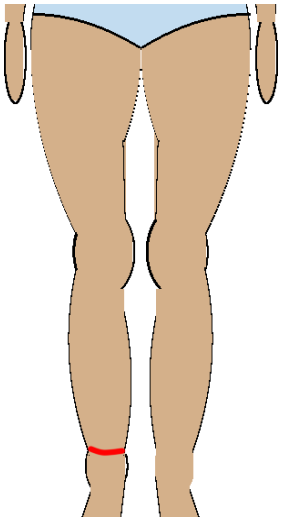


Рисунок 1.16 - Обхват щиколотки

Довжина рукава до ліктя: мірка вимірюється одночасно з міркою довжини рукава по злегка зігнутою в лікті руці від крайньої точки плеча до ліктя (рисунок 1.17).

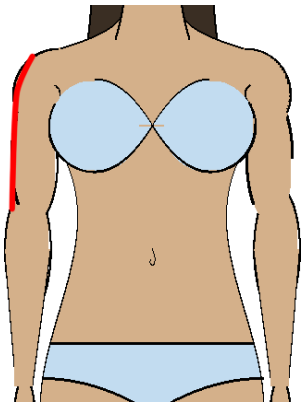


Рисунок 1.17 - Довжина рукава до ліктя

Обхват плеча: вимірюють навколо руки у пахвовій западини (рисунок 1.18)

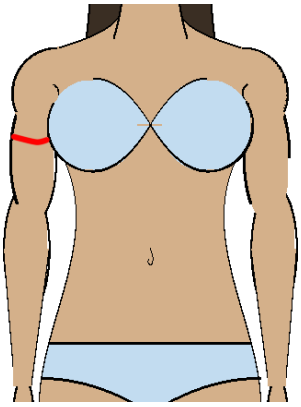


Рисунок 1.18 - Обхват плеча

Довжина рукава: вимірюють від крайньої точки плеча до кисті руки по злегка зігнутій в лікті руці (рисунок 1.19).

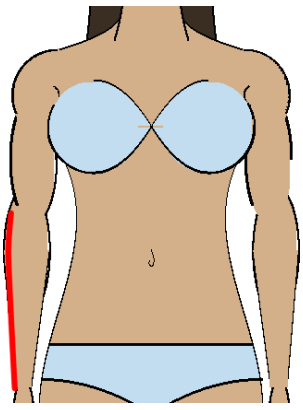


Рисунок 1.19 - Довжина рукава

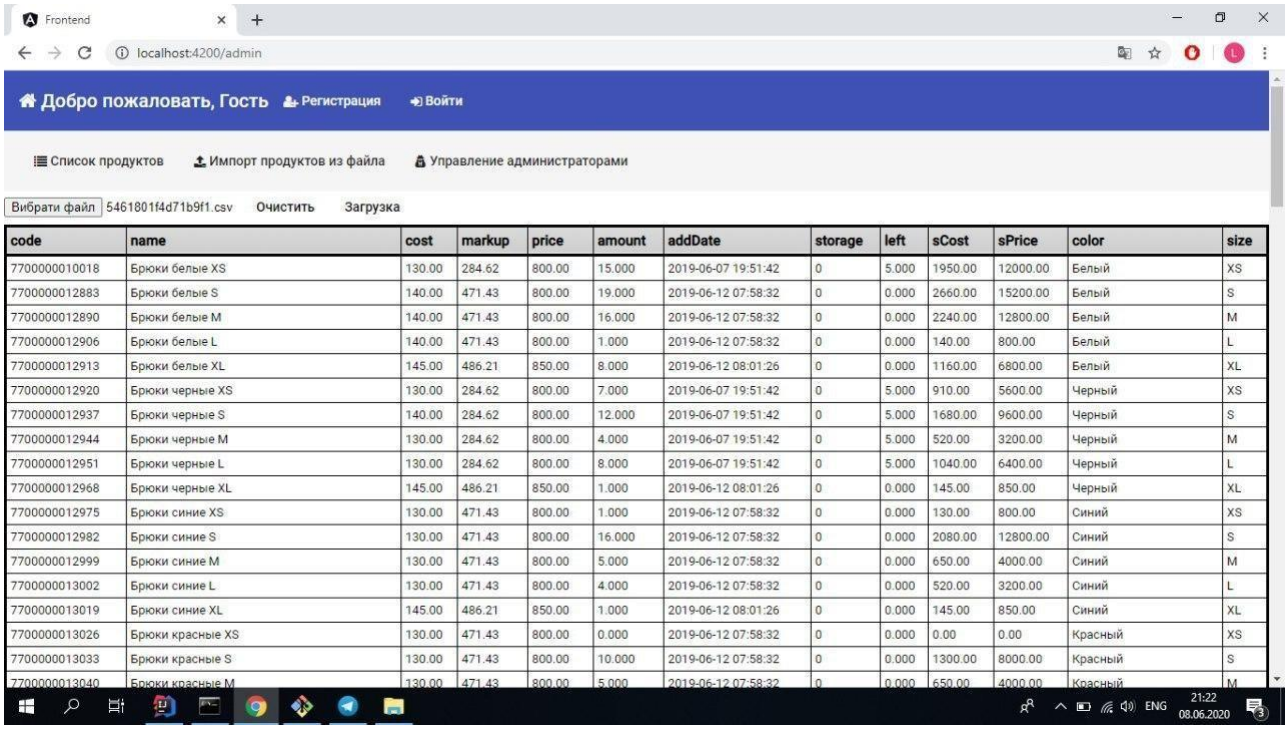
Обхват зап’ястя: вимірюють навколо руки по лучезапястному суглобу (рисунок 1.20)

Рисунок 1.20 - Обхват зап'ястя

2 ІНСТРУКЦІЯ АДМІНІСТРАТОРА

2.1 Імпорт з програми обліку товарів

На сторінці адміністратора потрібно натиснути імпорт продуктів з файлу, обрати файл, та натиснути завантаження (рисунок 2.1).



code	name	cost	markup	price	amount	addDate	storage	left	sCost	sPrice	color	size
7700000010018	Брюки белые XS	130.00	284.62	800.00	15.000	2019-06-07 19:51:42	0	5.000	1950.00	12000.00	Белый	XS
7700000012883	Брюки белые S	140.00	471.43	800.00	19.000	2019-06-12 07:58:32	0	0.000	2660.00	15200.00	Белый	S
7700000012890	Брюки белые M	140.00	471.43	800.00	16.000	2019-06-12 07:58:32	0	0.000	2240.00	12800.00	Белый	M
7700000012906	Брюки белые L	140.00	471.43	800.00	1.000	2019-06-12 07:58:32	0	0.000	140.00	800.00	Белый	L
7700000012913	Брюки белые XL	145.00	486.21	850.00	8.000	2019-06-12 08:01:26	0	0.000	1160.00	6800.00	Белый	XL
7700000012920	Брюки черные XS	130.00	284.62	800.00	7.000	2019-06-07 19:51:42	0	5.000	910.00	5600.00	Черный	XS
7700000012937	Брюки черные S	140.00	284.62	800.00	12.000	2019-06-07 19:51:42	0	5.000	1680.00	9600.00	Черный	S
7700000012944	Брюки черные M	130.00	284.62	800.00	4.000	2019-06-07 19:51:42	0	5.000	520.00	3200.00	Черный	M
7700000012951	Брюки черные L	130.00	284.62	800.00	8.000	2019-06-07 19:51:42	0	5.000	1040.00	6400.00	Черный	L
7700000012968	Брюки черные XL	145.00	486.21	850.00	1.000	2019-06-12 08:01:26	0	0.000	145.00	850.00	Черный	XL
7700000012975	Брюки синие XS	130.00	471.43	800.00	1.000	2019-06-12 07:58:32	0	0.000	130.00	800.00	Синий	XS
7700000012982	Брюки синие S	130.00	471.43	800.00	16.000	2019-06-12 07:58:32	0	0.000	2080.00	12800.00	Синий	S
7700000012999	Брюки синие M	130.00	471.43	800.00	5.000	2019-06-12 07:58:32	0	0.000	650.00	4000.00	Синий	M
7700000013002	Брюки синие L	130.00	471.43	800.00	4.000	2019-06-12 07:58:32	0	0.000	520.00	3200.00	Синий	L
7700000013019	Брюки синие XL	145.00	486.21	850.00	1.000	2019-06-12 08:01:26	0	0.000	145.00	850.00	Синий	XL
7700000013026	Брюки красные XS	130.00	471.43	800.00	0.000	2019-06-12 07:58:32	0	0.000	0.00	0.00	Красный	XS
7700000013033	Брюки красные S	130.00	471.43	800.00	10.000	2019-06-12 07:58:32	0	0.000	1300.00	8000.00	Красный	S
7700000013040	Брюки красные M	130.00	471.43	800.00	5.000	2019-06-12 07:58:32	0	0.000	650.00	4000.00	Красный	M

Рисунок 2.1 – Імпорт продуктів з файлу

ВФакультет інформатики та обчислювальної техніки

Кафедра автоматизованих систем обробки інформації і управління

“ЗАТВЕРДЖЕНО”

В.о. завідувача кафедри

_____Олександр ПАВЛОВ

“___”_____2020 р.

ВЕБ-ЗАСТОСУВАННЯ «ВІРТУАЛЬНЕ АТЕЛЬЄ»

Опис програми

КПІ.ІП-6315.045440.06.13

“ПОГОДЖЕНО”

Керівник проєкту:

_____О.А. Халус

Нормоконтроль:

_____К.І. Ліщук

Виконавець:

_____Л.В. Корольова

Київ – 2020 року

Тексти програмного коду

Веб-застосування «Віртуальне ательє»

(Найменування програми (документа))

DVD-R

(Вид носія даних)

34 арк, 246 Кб

(Обсяг програми (документа) , арк.,) Кб)

Київ - 2020

body.py

```
from convert import *
from colors import *
from figures import *

class BodyBuilder:
    def __init__(self,img,skinColor):
        self.img = img
        self.Xcentr = img.shape[1]/2
        self.Ycentr = img.shape[0]/2
        self.SKIN = skinColor

    def DrowHead(self):
        R=75
        DrowCircle(self.img,R,self.Xcentr,200,BROWN,BORDERCOLOR=BLACK,st
art=mth.pi-mth.pi/6,end=mth.pi*2+mth.pi/6)

        points=[]
        points.append([self.Xcentr+R*mth.cos(mth.pi-
mth.pi/6),200+R*mth.sin(mth.pi-mth.pi/6)])
        points.append([self.Xcentr,300])
        points.append([self.Xcentr+R*mth.cos(mth.pi*2+mth.pi/6),200+R*mt
h.sin(mth.pi*2+mth.pi/6)])
        borders=[]
        borders.append([self.Xcentr-R,200])
        borders.append([self.Xcentr+R,300])

        DrowElipse(img = self.img,
                   a = R,
                   b = R *2/3,
                   X = self.Xcentr,
                   Y = 200,
                   COLOR = self.SKIN,
                   BORDERCOLOR = BLACK,
                   start = mth.pi-mth.pi/6,
                   end = mth.pi*2+mth.pi/6,
                   overlay=True)

        a,b,c = DrowParabola(self.img,points,borders,self.SKIN)
        return a,b,c

    def DrowNeck(self,circumference):
        widthSm = circumference/mth.pi
        widthPixel = FromSmToPixel(widthSm)
        a2,b2 =
DrowHiperbolaByBParam(self.img,widthPixel,self.Xcentr,290,4500,250,350,s
elf.SKIN,overlay=False)
        x1,x2 =
GetHiperbolaPointByBParam(widthPixel,self.Xcentr,290,4500,320)
        return [[x1,320],[x2,320]]

    def DrowShoulders(self,length,neck):
        l = FromSmToPixel(length)
```

```
        x2 = neck[1][0]
        y2 = neck[1][1]

        y1 = 360
        delta=(l*l-40*40)**0.5
        x1=x2-delta
        shoulderPoints=[]
        shoulderPoints.append([x1,y1])
        DrowBrunch(self.img,[[x1,y1],[x2,y2]],self.SKIN,overlay=False)
        x1 = neck[0][0]
        x2 = x1+delta
        shoulderPoints.append([x2,y1])
        DrowBrunch(self.img,[[x1,y2],[x2,y1]],self.SKIN,overlay=False)

        R=75
        DrowHiperbolaByPoints(img = self.img,
                              width = (neck[1][0]-neck[0][0])*2,
                              X = self.Xcentr,
                              Y = 290,
                              point = [self.Xcentr+R,200],
                              YB = 200+R*mth.sin(mth.pi-mth.pi/6),
                              YE = 350,
                              COLOR = BROWN,
                              BORDERCOLOR=BLACK,
                              overlay=False)
        DrowRectangle(img = self.img,
                      centr = [self.Xcentr,y2 + (y1-y2)/2],
                      halfA = x1 - self.Xcentr,
                      halfB = (y1-y2)/2,
                      COLOR = self.SKIN,
                      overlay=False)
        return shoulderPoints

    def DrowWomanChest(self, chestCircum, underChestCircum, backWidth,
chestHight, chestCentr, neckPoints, shoulderPoints):
        leftPoint = neckPoints[0]
        rightPoint = neckPoints[1]

        chest = FromSmToPixel(chestCircum)
        underChest = FromSmToPixel(underChestCircum)
        back = FromSmToPixel(backWidth)
        hight = FromSmToPixel(chestHight)
        centr = FromSmToPixel(chestCentr)

        k = (centr -rightPoint[0]+leftPoint[0])/2
        h = (hight*hight-k*k)**0.5

        LeftChestPoint=[leftPoint[0]-k,leftPoint[1]+h]
        RightChestPoint=[rightPoint[0]+k,leftPoint[1]+h]

        k=0.6
        width = (underChest*(k+1))/(4*mth.pi*k+(k-1)**2)
        leftX = self.Xcentr-width
        rightX = self.Xcentr+width
        R,rbAngle,reAngle = GetChestSizeFromSize(chestCircum-
underChestCircum)
```

```
        lbAngle = 3*mth.pi-reAngle
        leAngle = 3*mth.pi-rbAngle

        DrowCircle(self.img,R,RightChestPoint[0],RightChestPoint[1],BLUE
,start=rbAngle,end=reAngle)
        DrowCircle(self.img,R,LeftChestPoint[0],LeftChestPoint[1],BLUE,s
tart=lbAngle,end=leAngle)
        DrowParabola(self.img,[[self.Xcentr,RightChestPoint[1]],
                               [RightChestPoint[0],RightChestPoint[1]-R],
                               [RightChestPoint[0]*2-
self.Xcentr,LeftChestPoint[1]]],
                     [[self.Xcentr,RightChestPoint[1]-R],
                      [RightChestPoint[0],RightChestPoint[1]]],
                     BLUE,part=False)
        DrowParabola(self.img,[[self.Xcentr,RightChestPoint[1]],
                               [RightChestPoint[0],RightChestPoint[1]+R],
                               [RightChestPoint[0]*2-
self.Xcentr,LeftChestPoint[1]]],
                     [[self.Xcentr,RightChestPoint[1]],
                      [RightChestPoint[0],RightChestPoint[1]+R]],
                     BLUE)

        DrowParabola(self.img,[[LeftChestPoint[0]*2-
self.Xcentr,LeftChestPoint[1]],
                               [LeftChestPoint[0],LeftChestPoint[1]-R],
                               [self.Xcentr,LeftChestPoint[1]]],
                     [[LeftChestPoint[0],LeftChestPoint[1]-R],
                      [self.Xcentr,LeftChestPoint[1]]],
                     BLUE,part=False)

        DrowParabola(self.img,[[LeftChestPoint[0]*2-
self.Xcentr,LeftChestPoint[1]],
                               [LeftChestPoint[0],LeftChestPoint[1]+R],
                               [self.Xcentr,LeftChestPoint[1]]],
                     [[LeftChestPoint[0],LeftChestPoint[1]],
                      [self.Xcentr,LeftChestPoint[1]+R]],
                     BLUE)

        y=GetCircleYByX(R,LeftChestPoint[0],LeftChestPoint[1],leftX)
        LeftUnderChestPoint = [leftX,y[1]]
        y=GetCircleYByX(R,RightChestPoint[0],RightChestPoint[1],rightX)
        RightUnderChestPoint = [rightX,y[1]]

        point=GetCirclePointByAngle(R,LeftChestPoint[0],LeftChestPoint[1
],leAngle)
        DrowHiperbola(self.img,point,self.Xcentr,point[1]-
10,1000,point[1]-20,point[1],self.SKIN,overlay=False)

        DrowRectangle(img = self.img,
                      centr = [self.Xcentr,LeftChestPoint[1]+R/2],
                      halfA = width/2,
                      halfB = R/2,
                      COLOR = self.SKIN,
                      overlay = False)

        DrowRectangle(img = self.img,
```

```
                        centr =
[self.Xcentr,shoulderPoints[0][1]+(LeftChestPoint[1]-
shoulderPoints[0][1])/2],
                        halfA = self.Xcentr-shoulderPoints[0][0],
                        halfB = (LeftChestPoint[1]-
shoulderPoints[0][1])/2,
                        COLOR = self.SKIN,
                        overlay = False)

        chestPoints=[LeftChestPoint,RightChestPoint]
        underChestPoints=[LeftUnderChestPoint,RightUnderChestPoint]

        return chestPoints,underChestPoints

    def
DrowAbdomen(self,waist,waistHight,hips,hipLength,hip,underChestPoints,ne
ckPoints):
        pixelWaist = FromSmToPixel(waist)
        pixelHight = FromSmToPixel(waistHight)
        pixelHips = FromSmToPixel(hips)
        pixelHip = FromSmToPixel(hip)
        pixelLength = FromSmToPixel(hipLength)

        #waist point
        k=0.6
        a1 = (pixelWaist*(k+1))/(4*mth.pi*k+(k-1)**2)
        x = self.Xcentr - a1
        y = neckPoints[0][1]+pixelHight
        waistPoint=[x,y]
        #hips point
        k=0.6
        a2 = (pixelHips*(k+1))/(4*mth.pi*k+(k-1)**2)
        x = self.Xcentr - a2
        h = (pixelLength**2-(a2-a1)**2)**0.5
        y += h
        hipsPoint = [x,y]
        #hip point
        k=0.8
        a3 = 2*(pixelHip*(k+1))/(4*mth.pi*k+(k-1)**2)
        x = self.Xcentr - a3
        y+=h*0.5
        hipPoint = [x,y]
        #drow

        params=DrowDoublePolynom(img = self.img,
                                 points =
[underChestPoints[0],waistPoint,hipsPoint,hipPoint],
                                 X = self.Xcentr,
                                 COLOR = self.SKIN,
                                 overlay=False)

        DrowDoublePolynomByParams(img = self.img,
                                  params = params,
                                  X = self.Xcentr,
                                  COLOR = BLUE,
```

```
                                    borders = [hipsPoint[1]-
h/6,hipsPoint[1]+h/6])

        x1=GetPolynomXByyandParam(hipsPoint[1]+h/6,params)
        b=GetPolynomXByyandParam(hipsPoint[1]-h/6,params)
        DrowParabola(self.img,[[x1-
a3,y],[x1,hipsPoint[1]+h/6],[self.Xcentr,y]],[[x1,hipsPoint[1]+h/6],[sel
f.Xcentr,y]],BLUE)
        x1=2 * self.Xcentr - x1
        e=2 * self.Xcentr - b
        DrowParabola(self.img,[[self.Xcentr,y],[x1,hipsPoint[1]+h/6],[2*
x1-self.Xcentr,y]],[[self.Xcentr,hipsPoint[1]+h/6],[x+2*a3,y]],BLUE)


        DrowLine(self.img,[b,hipsPoint[1]-h/6],[e,hipsPoint[1]-h/6])
        self.DrowNavel(self.Xcentr,neckPoints[0][1]+pixelHight)

        hipPoints = []
        hipPoints.append(hipPoint)
        hipPoints.append([self.Xcentr + a3,hipPoint[1]])

        return hipPoints

    def DrowNavel(self,X,Y):
        X=int(X)
        Y=int(Y)
        self.img[Y+5,X-3]=BLACK
        self.img[Y+6,X-3]=BLACK

        self.img[Y+3,X-2]=BLACK
        self.img[Y+4,X-2]=BLACK
        self.img[Y+7,X-2]=BLACK
        self.img[Y+8,X-2]=BLACK

        self.img[Y+8,X-1]=BLACK
        self.img[Y+9,X-1]=BLACK

        self.img[Y-9,X]=BLACK
        self.img[Y-8,X]=BLACK
        self.img[Y-7,X]=BLACK
        self.img[Y+9,X]=BLACK

        self.img[Y-6,X+1]=BLACK
        self.img[Y-5,X+1]=BLACK
        self.img[Y-4,X+1]=BLACK
        self.img[Y+8,X+1]=BLACK
        self.img[Y+9,X+1]=BLACK

        self.img[Y-3,X+2]=BLACK
        self.img[Y-2,X+2]=BLACK
        self.img[Y-1,X+2]=BLACK
        self.img[Y,X+2]=BLACK
        self.img[Y+5,X+2]=BLACK
        self.img[Y+6,X+2]=BLACK
        self.img[Y+7,X+2]=BLACK
        self.img[Y+8,X+2]=BLACK
```

```
        self.img[Y+1,X+3]=BLACK
        self.img[Y+2,X+3]=BLACK
        self.img[Y+3,X+3]=BLACK
        self.img[Y+4,X+3]=BLACK

    def DrowHands(self,firstHandCircum,
secondHandCircum,firstHandLength,secondHandLength,shoulderPoints):
        Rsm = firstHandCircum/mth.pi
        R = FromSmToPixel(Rsm)

        l1 = FromSmToPixel(firstHandLength)
        l2 = FromSmToPixel(secondHandLength)
        l3 = FromSmToPixel(secondHandCircum)/mth.pi
        a=R/2
        b=(l1-R)/2
        #leftHand
        Xleft = shoulderPoints[0][0] + R*mth.sin(mth.pi/4)
        Yleft = shoulderPoints[0][1] + R*mth.sin(mth.pi/4)
        DrowSector(img = self.img,
                   R = R,
                   X = Xleft,
                   Y = Yleft,
                   COLOR = self.SKIN,
                   start = 1*mth.pi-mth.pi/8,
                   end = 1.5*mth.pi-mth.pi/4,
                   overlay = False)

        DrowSector(img = self.img,
                   R = R,
                   X = Xleft,
                   Y = Yleft,
                   COLOR = self.SKIN,
                   BORDERCOLOR = self.SKIN,
                   start = 0.6*mth.pi,
                   end = 1*mth.pi-mth.pi/8,
                   overlay = False)

        DrowElipseConture(img = self.img,
                   a = a,
                   b = b,
                   X = Xleft-a*1.1,
                   Y = Yleft+b,
                   BORDERCOLOR = BLACK,
                   start = mth.pi/2+mth.pi/4,
                   end = mth.pi*2,
                   overlay=False)

        DrowElipse(img = self.img,
                   a = a,
                   b = b,
                   X = Xleft-a*1.1,
                   Y = Yleft+b,
                   COLOR = self.SKIN,
                   BORDERCOLOR = BLACK,
                   start = 0-mth.pi/2,
                   end = mth.pi/3,
```

```
                    overlay=False)
            b2 = l2/3
            leftPoint = GetElipsePointByAngle(a = a,
                                              b = b,
                                              X = Xleft-a*1.4,
                                              Y = Yleft+b+b+b2*0.6,
                                              angle = mth.pi)

            rightPoint = GetElipsePointByAngle(a = a,
                                              b = b,
                                              X = Xleft-a*1.4,
                                              Y = Yleft+b+b+b2*0.6,
                                              angle = 0)
            downRightPoint = [rightPoint[0]-
l2*mth.sin(mth.pi/12),rightPoint[1]+l2*mth.cos(mth.pi/12)]
            DrowColorLine(img = self.img,
                          point1 = rightPoint,
                          point2 = downRightPoint,
                          COLOR = self.SKIN,
                          part = True,
                          BORDERCOLOR=BLACK,
                          overlay=True)

            DrowColorLine(img = self.img,
                          point1 = leftPoint,
                          point2 = [downRightPoint[0]-
l3*2/3,downRightPoint[1]],
                          COLOR = self.SKIN,
                          part = False,
                          BORDERCOLOR=BLACK,
                          overlay=True)

            DrowRectangle(img = self.img,
                          centr = [downRightPoint[0]-
l3/3,rightPoint[1]+(downRightPoint[1]-rightPoint[1])/2],
                          halfA = l3/3,
                          halfB = (downRightPoint[1]-rightPoint[1])/2,
                          COLOR = self.SKIN,
                          overlay = True)

            DrowElipse(img = self.img,
                       a = a,
                       b = b,
                       X = Xleft-a*1.4,
                       Y = Yleft+b+b+b2*0.6,
                       COLOR = self.SKIN,
                       BORDERCOLOR = BLACK,
                       overlay=False)

            DrowElipse(img = self.img,
                       a = 1.3*l3/3,
                       b = l2/3,
                       X = downRightPoint[0]-l3/3,
                       Y = downRightPoint[1]+0.8*l2/3,
                       COLOR = self.SKIN,
                       BORDERCOLOR = BLACK,
                       overlay=False)
```

```
        #rightHand
        DrowSector(self.img,
                   R = R,
                   X =  shoulderPoints[1][0],
                   Y = shoulderPoints[1][1]+R,
                   COLOR = self.SKIN,
                   start = 1.5*mth.pi,
                   end = 2*mth.pi+mth.pi/8,
                   overlay = False)
        DrowSector(self.img,
                   R = R,
                   X =  shoulderPoints[1][0],
                   Y = shoulderPoints[1][1]+R,
                   COLOR = self.SKIN,
                   BORDERCOLOR = self.SKIN,
                   start = 2*mth.pi+mth.pi/8,
                   end = 2.4*mth.pi,
                   overlay = False)

        DrowElipseConture(img = self.img,
                   a = a,
                   b = b,
                   X = shoulderPoints[1][0]+a*1.1,
                   Y = shoulderPoints[1][1]+R+b,
                   BORDERCOLOR = BLACK,
                   start = mth.pi/2+mth.pi/4,
                   end = mth.pi*2,
                   overlay=False)

        DrowElipse(img = self.img,
                   a = a,
                   b = b,
                   X = shoulderPoints[1][0]+a*1.1,
                   Y = shoulderPoints[1][1]+R+b,
                   COLOR = self.SKIN,
                   BORDERCOLOR = BLACK,
                   start = mth.pi+mth.pi/2,
                   end = mth.pi-mth.pi/3,
                   overlay=False)
        b2 = l2/3
        leftPoint = GetElipsePointByAngle(a = a,
                                          b = b,
                                          X =
shoulderPoints[1][0]+a*1.4,
                                          Y =
shoulderPoints[1][1]+R+b+b+b2*0.6,
                                          angle = mth.pi)

        rightPoint = GetElipsePointByAngle(a = a,
                                          b = b,
                                          X =
shoulderPoints[1][0]+a*1.4,
                                          Y =
shoulderPoints[1][1]+R+b+b+b2*0.6,
                                          angle = 0)
```

```
        downLeftPoint =
[leftPoint[0]+l2*mth.sin(mth.pi/24),leftPoint[1]+l2*mth.cos(mth.pi/24)]
        DrowColorLine(img = self.img,
                      point1 = leftPoint,
                      point2 = downLeftPoint,
                      COLOR = self.SKIN,
                      part = False,
                      BORDERCOLOR=BLACK,
                      overlay=True)

        DrowColorLine(img = self.img,
                      point1 = rightPoint,
                      point2 =
[downLeftPoint[0]+l3*2/3,downLeftPoint[1]],
                      COLOR = self.SKIN,
                      part = True,
                      BORDERCOLOR=BLACK,
                      overlay=True)

        DrowRectangle(img = self.img,
                      centr =
[downLeftPoint[0]+l3/3,leftPoint[1]+(downLeftPoint[1]-leftPoint[1])/2],
                      halfA = l3/3,
                      halfB = (downLeftPoint[1]-leftPoint[1])/2,
                      COLOR = self.SKIN,
                      overlay = True)

        DrowElipse(img = self.img,
                   a = a,
                   b = b,
                   X = shoulderPoints[1][0]+a*1.4,
                   Y = shoulderPoints[1][1]+R+b+b+b2*0.6,
                   COLOR = self.SKIN,
                   BORDERCOLOR = BLACK,
                   overlay=False)

        DrowElipse(img = self.img,
                   a = 1.3*l3/3,
                   b = l2/3,
                   X = downLeftPoint[0]+l3/3,
                   Y = downLeftPoint[1]+0.8*l2/3,
                   COLOR = self.SKIN,
                   BORDERCOLOR = BLACK,
                   overlay=False)

    def DrowLegs(self,legLength,topCircum,
kneeCircum,calfCircum,buttomCircum,hipPoints):

        leg = FromSmToPixel(legLength)
        top = FromSmToPixel(topCircum)
        knee = FromSmToPixel(kneeCircum)
        calf = FromSmToPixel(calfCircum)
        buttom = FromSmToPixel(buttomCircum)

        kneeR = knee/(2*mth.pi)
        leftX = hipPoints[0][0]+(hipPoints[1][0]-hipPoints[0][0])/3
        rightX = hipPoints[0][0]+2*(hipPoints[1][0]-hipPoints[0][0])/3
```

```
        calfR = calf/(2*mth.pi)
        buttomR = buttom/(2*mth.pi)

        #leftLeg
        DrowElipse(img = self.img,
                   a = kneeR,
                   b = kneeR * 1.5,
                   X = leftX,
                   Y = hipPoints[0][1]+leg * 0.45,
                   COLOR = self.SKIN,
                   BORDERCOLOR = BLACK,
                   start = 0-mth.pi/4,
                   end = mth.pi/3,
                   overlay=True)

        DrowElipseConture(img = self.img,
                   a = kneeR,
                   b = kneeR * 1.5,
                   X = leftX,
                   Y = hipPoints[0][1]+leg * 0.45,
                   BORDERCOLOR = BLACK,
                   start = mth.pi-mth.pi/8,
                   end = mth.pi+mth.pi/8,
                   overlay=True)


        leftPoint = GetElipsePointByAngle(a = kneeR,
                                          b = kneeR*1.5,
                                          X = leftX,
                                          Y = hipPoints[0][1]+leg *
0.45,
                                          angle = mth.pi+mth.pi/8)

        leftCentrPoint = GetElipsePointByAngle(a = kneeR,
                                               b = kneeR*1.5,
                                               X = leftX,
                                               Y = hipPoints[0][1]+leg *
0.45,
                                               angle = 0-mth.pi/4)

        p2 = DrowBrunch(img = self.img,
                   points = [hipPoints[0],leftPoint],
                   COLOR = self.SKIN,
                   BORDERCOLOR=BLACK,
                   part = False,
                   overlay=True)

        halfA = (leftCentrPoint[0]-leftPoint[0])/2
        halfB = (leftPoint[1]-hipPoints[0][1])/2
        xLeft =
GetBrunchPointByY(p2,hipPoints[1][1]+halfB,hipPoints[0][0],hipPoints[0][
1])
        xLeftCentr = xLeft+top/mth.pi

        DrowBrunch(img = self.img,
```

```
                    points =
[[xLeftCentr,hipPoints[1][1]+halfB],[self.Xcentr,hipPoints[0][1]]],
                    COLOR = self.SKIN,
                    BORDERCOLOR=BLACK,
                    part = False,
                    overlay=True)

        DrowLine(img = self.img,
                 point1 = leftCentrPoint,
                 point2 = [xLeftCentr,hipPoints[1][1]+halfB],
                 BORDERCOLOR=BLACK,
                 overlay=True)

        DrowRectangle(img = self.img,
                      centr =
[leftPoint[0]+halfA,hipPoints[0][1]+halfB],
                      halfA = halfA,
                      halfB = halfB,
                      COLOR = self.SKIN,
                      overlay = False)

        centr = [leftX-kneeR*0.2,hipPoints[0][1]+leg * 0.62]

        first = GetElipsePointByAngle(a = kneeR,
                                      b = kneeR*1.5,
                                      X = leftX,
                                      Y = hipPoints[0][1]+leg *
0.45,
                                      angle = mth.pi-mth.pi/8)
        second = [centr[0]-calfR,centr[1]]
        third = [self.Xcentr-buttomR*3,hipPoints[0][1]+leg * 0.9]
        leftPoints = [first,second,third]
        first = GetElipsePointByAngle(a = kneeR,
                                      b = kneeR*1.5,
                                      X = leftX,
                                      Y = hipPoints[0][1]+leg *
0.45,
                                      angle = mth.pi/3)
        second = [centr[0]+calfR,centr[1]]
        third = [self.Xcentr-buttomR,hipPoints[0][1]+leg * 0.9]
        rightPoints = [first,second,third]
        Drow2Polynom(img = self.img,
                     points = leftPoints,
                     COLOR = self.SKIN,
                     BORDERCOLOR = BLACK,
                     part=False,
                     overlay=True)

        Drow2Polynom(img = self.img,
                     points = rightPoints,
                     COLOR = self.SKIN,
                     BORDERCOLOR = BLACK,
                     part=True,
                     overlay=True)


        DrowRectangle(img = self.img,
```

```
                        centr = [self.Xcentr-2*buttomR,first[1]+(third[1]-
first[1])/2],
                        halfA = buttomR,
                        halfB = (third[1]-first[1])/2,
                        COLOR = self.SKIN,
                        overlay = True)

        #fit
        DrowElipse(img = self.img,
                   a = buttomR*1.2,
                   b = buttomR*1.2 * 1.5,
                   X = self.Xcentr - 2*buttomR,
                   Y = hipPoints[0][1]+leg * 0.925,
                   COLOR = self.SKIN,
                   BORDERCOLOR = BLACK,
                   start = 0-mth.pi/8,
                   end = mth.pi/8,
                   overlay=False)

        DrowElipseConture(img = self.img,
                   a = buttomR*1.2,
                   b = buttomR*1.2 * 1.5,
                   X = self.Xcentr - 2*buttomR,
                   Y = hipPoints[0][1]+leg * 0.925,
                   BORDERCOLOR = BLACK,
                   start = mth.pi-mth.pi/4,
                   end = mth.pi+mth.pi/3,
                   overlay=False)

        point1 = GetElipsePointByAngle(a = buttomR*1.2,
                                       b = buttomR*1.2 * 1.5,
                                       X = self.Xcentr - 2*buttomR,
                                       Y = hipPoints[0][1]+leg *
0.925,
                                       angle = mth.pi-mth.pi/4)

        DrowColorLine(img = self.img,
                  point1 = point1,
                  point2 = [self.Xcentr -
4*buttomR,hipPoints[0][1]+leg*1.1],
                  COLOR = self.SKIN,
                  part = False,
                  BORDERCOLOR=BLACK,
                  overlay=True)

        point2 = GetElipsePointByAngle(a = buttomR*1.2,
                                       b = buttomR*1.2 * 1.5,
                                       X = self.Xcentr - 2*buttomR,
                                       Y = hipPoints[0][1]+leg *
0.925,
                                       angle = mth.pi/8)

        DrowColorLine(img = self.img,
                  point1 = point2,
                  point2 = [self.Xcentr -
1.5*buttomR,hipPoints[0][1]+leg*1.15],
                  COLOR = self.SKIN,
```

```
                part = True,
                BORDERCOLOR=BLACK,
                overlay=True)


        DrowParabola(img = self.img,
                     points = [[self.Xcentr -
4*buttomR,hipPoints[0][1]+leg*1.1],
                               [self.Xcentr -
1.5*buttomR,hipPoints[0][1]+leg*1.15],
                               [self.Xcentr,hipPoints[0][1]+leg*1.1]],
                     borders = [[self.Xcentr -
4*buttomR,hipPoints[0][1]+leg*1.1],
                                [self.Xcentr -
1.5*buttomR,hipPoints[0][1]+leg*1.15]],
                     COLOR = self.SKIN,
                     BORDERCOLOR=BLACK,
                     part=True,
                     overlay=True)

        DrowRectangle(img = self.img,
                      centr = [point1[0]+(self.Xcentr - 1.5*buttomR-
point1[0])/2,point1[1]+(hipPoints[0][1]+leg*1.1-point1[1])/2],
                      halfA = (self.Xcentr - 1.5*buttomR-point1[0])/2,
                      halfB = (hipPoints[0][1]+leg*1.1-point1[1])/2,
                      COLOR = self.SKIN,
                      overlay = True)

        #right leg
        DrowElipse(img = self.img,
                   a = kneeR,
                   b = kneeR * 1.5,
                   X = rightX,
                   Y = hipPoints[0][1]+leg * 0.45,
                   COLOR = self.SKIN,
                   BORDERCOLOR = BLACK,
                   start = 0-mth.pi/8,
                   end = mth.pi/8,
                   overlay=True)

        DrowElipseConture(img = self.img,
                   a = kneeR,
                   b = kneeR * 1.5,
                   X = rightX,
                   Y = hipPoints[0][1]+leg * 0.45,
                   BORDERCOLOR = BLACK,
                   start = mth.pi-mth.pi/3,
                   end = mth.pi+mth.pi/4,
                   overlay=True)

        rightPoint = GetElipsePointByAngle(a = kneeR,
                                           b = kneeR*1.5,
                                           X = rightX,
                                           Y = hipPoints[0][1]+leg *
0.45,
                                           angle = 0-mth.pi/8)
```

```
        rightCentrPoint = GetElipsePointByAngle(a = kneeR,
                                                b = kneeR*1.5,
                                                X = rightX,
                                                Y = hipPoints[0][1]+leg
* 0.45,
                                                angle = mth.pi+mth.pi/4)

        p2=DrowBrunch(img = self.img,
                      points = [rightPoint,hipPoints[1]],
                      COLOR = self.SKIN,
                      BORDERCOLOR=BLACK,
                      part = False,
                      overlay=True)

        xRight =
GetBrunchPointByY(p2,hipPoints[1][1]+halfB,hipPoints[1][0],hipPoints[1][
1])
        xRightCentr = xRight-top/mth.pi

        DrowBrunch(img = self.img,
                   points =
[[self.Xcentr,hipPoints[0][1]],[xRightCentr,hipPoints[1][1]+halfB]],
                   COLOR = self.SKIN,
                   BORDERCOLOR=BLACK,
                   part = False,
                   overlay=True)

        DrowLine(img = self.img,
                 point1 = rightCentrPoint,
                 point2 = [xRightCentr,hipPoints[1][1]+halfB],
                 BORDERCOLOR=BLACK,
                 overlay=True)

        DrowRectangle(img = self.img,
                      centr = [rightPoint[0]-
halfA,hipPoints[0][1]+halfB],
                      halfA = halfA,
                      halfB = halfB,
                      COLOR = self.SKIN,
                      overlay = False)

        rightKneePoint = GetElipsePointByAngle(a = kneeR,
                                               b = kneeR*1.5,
                                               X = rightX,
                                               Y = hipPoints[0][1]+leg *
0.45,
                                               angle =  mth.pi/8)

        buttomPoint = [self.Xcentr+2.5*buttomR,hipPoints[0][1]+leg*0.9]

        centr = [rightX+kneeR*0.2,hipPoints[0][1]+leg * 0.62]

        first = GetElipsePointByAngle(a = kneeR,
                                               b = kneeR*1.5,
                                               X = rightX,
                                               Y = hipPoints[0][1]+leg *
0.45,
```

```
                                        angle = mth.pi/8)
        second = [centr[0]+calfR,centr[1]]
        third = [self.Xcentr+buttomR*3,hipPoints[0][1]+leg * 0.9]
        rightPoints = [first,second,third]
        first = GetElipsePointByAngle(a = kneeR,
                                        b = kneeR*1.5,
                                        X = rightX,
                                        Y = hipPoints[0][1]+leg *
0.45,
                                        angle = mth.pi-mth.pi/3)
        second = [centr[0]-calfR,centr[1]]
        third = [self.Xcentr+buttomR,hipPoints[0][1]+leg * 0.9]
        leftPoints = [first,second,third]
        Drow2Polynom(img = self.img,
                     points = leftPoints,
                     COLOR = self.SKIN,
                     BORDERCOLOR = BLACK,
                     part=False,
                     overlay=True)

        Drow2Polynom(img = self.img,
                     points = rightPoints,
                     COLOR = self.SKIN,
                     BORDERCOLOR = BLACK,
                     part=True,
                     overlay=True)


        DrowRectangle(img = self.img,
                      centr = [self.Xcentr+2*buttomR,first[1]+(third[1]-
first[1])/2],
                      halfA = buttomR,
                      halfB = (third[1]-first[1])/2,
                      COLOR = self.SKIN,
                      overlay = True)

               #fit
        DrowElipse(img = self.img,
                   a = buttomR*1.2,
                   b = buttomR*1.2 * 1.5,
                   X = self.Xcentr + 2*buttomR,
                   Y = hipPoints[0][1]+leg * 0.925,
                   COLOR = self.SKIN,
                   BORDERCOLOR = BLACK,
                   start = mth.pi+mth.pi/8,
                   end = mth.pi-mth.pi/8,
                   overlay=True)

        DrowElipseConture(img = self.img,
                   a = buttomR*1.2,
                   b = buttomR*1.2 * 1.5,
                   X = self.Xcentr + 2*buttomR,
                   Y = hipPoints[0][1]+leg * 0.925,
                   BORDERCOLOR = BLACK,
                   start = mth.pi/4,
                   end = 0 - mth.pi/3,
                   overlay=True)
```

```
        point1 = GetElipsePointByAngle(a = buttomR*1.2,
                                       b = buttomR*1.2 * 1.5,
                                       X = self.Xcentr + 2*buttomR,
                                       Y = hipPoints[0][1]+leg *
0.925,
                                       angle = mth.pi/4)

        DrowColorLine(img = self.img,
                  point1 = point1,
                  point2 = [self.Xcentr +
4*buttomR,hipPoints[0][1]+leg*1.1],
                  COLOR = self.SKIN,
                  part = True,
                  BORDERCOLOR=BLACK,
                  overlay=True)

        point2 = GetElipsePointByAngle(a = buttomR*1.2,
                                       b = buttomR*1.2 * 1.5,
                                       X = self.Xcentr + 2*buttomR,
                                       Y = hipPoints[0][1]+leg *
0.925,
                                       angle = mth.pi - mth.pi/8)

        DrowColorLine(img = self.img,
                  point1 = point2,
                  point2 = [self.Xcentr +
1.5*buttomR,hipPoints[0][1]+leg*1.15],
                  COLOR = self.SKIN,
                  part = False,
                  BORDERCOLOR=BLACK,
                  overlay=True)


        DrowParabola(img = self.img,
                     points = [[self.Xcentr,hipPoints[0][1]+leg*1.1],
                               [self.Xcentr +
1.5*buttomR,hipPoints[0][1]+leg*1.15],
                               [self.Xcentr +
4*buttomR,hipPoints[0][1]+leg*1.1]],
                     borders = [[self.Xcentr +
1.5*buttomR,hipPoints[0][1]+leg*1.1],
                                [self.Xcentr +
4*buttomR,hipPoints[0][1]+leg*1.15]],
                     COLOR = self.SKIN,
                     BORDERCOLOR=BLACK,
                     part=True,
                     overlay=True)

        DrowRectangle(img = self.img,
                      centr = [point1[0]-(point1[0]-self.Xcentr -
1.5*buttomR)/2,point1[1]+(hipPoints[0][1]+leg*1.1-point1[1])/2],
                      halfA = (point1[0]-self.Xcentr - 1.5*buttomR)/2,
                      halfB = (hipPoints[0][1]+leg*1.1-point1[1])/2,
                      COLOR = self.SKIN,
                      overlay = True)
```

figures.py

```
import numpy as np
import math as mth
from colors import *


def
DrowCircle(img,R,X,Y,COLOR,BORDERCOLOR=BLACK,start=0,end=2*mth.pi,overla
y=True):
    angle=start
    if(overlay):
        while(angle<end):
            x = R*mth.cos(angle)+X
            y = R*mth.sin(angle)+Y
            img[int(y)][int(x)]=BORDERCOLOR
            img[int(y)+1][int(x)]=BORDERCOLOR
            img[int(y)][int(x)+1]=BORDERCOLOR
            img[int(y)+1][int(x)+1]=BORDERCOLOR
            img[int(y)][int(x)-1]=BORDERCOLOR
            img[int(y)+1][int(x)]=BORDERCOLOR
            angle+=0.01
        x1=-R
        x2=+R
        y1=-R
        y2=+R
        for x in range(int(x1)+2,int(x2)-2):
            for y in range(int(y1)+2,int(y2)-2):
                if(x*x+y*y<R*R):
                    if(np.array_equal(img[int(y+Y)][int(x+X)],BORDERCOLO
R)==False):
                        img[int(y+Y)][int(x+X)]=COLOR
    else:
        while(angle<end):
            x = R*mth.cos(angle)+X
            y = R*mth.sin(angle)+Y
            if(np.array_equal(img[int(y)][int(x)],WHITE)):
                img[int(y)][int(x)]=BORDERCOLOR
            if(np.array_equal(img[int(y)+1][int(x)],WHITE)):
                img[int(y)+1][int(x)]=BORDERCOLOR
            if(np.array_equal(img[int(y)][int(x)+1],WHITE)):
                img[int(y)][int(x)+1]=BORDERCOLOR
            if(np.array_equal(img[int(y)+1][int(x)+1],WHITE)):
                img[int(y)+1][int(x)+1]=BORDERCOLOR
            if(np.array_equal(img[int(y)][int(x)-1],WHITE)):
                img[int(y)][int(x)-1]=BORDERCOLOR
            if(np.array_equal(img[int(y)+1][int(x)-1],WHITE)):
                img[int(y)+1][int(x)-1]=BORDERCOLOR
            angle+=0.01
        x1=-R
        x2=+R
        y1=-R
        y2=+R
        for x in range(int(x1)+1,int(x2)-1):
            for       y       in       range(int(y1)+1,int(y2)-1):
                if(np.array_equal(img[int(y+Y)][int(x+X)],WHITE)):
                    if(x*x+y*y<R*R):
```

```
                            img[int(y+Y)][int(x+X)]=COLOR
def
DrowSector(img,R,X,Y,COLOR,BORDERCOLOR=BLACK,start=0,end=2*mth.pi,overla
y=True):
    angle=start
    if(overlay):
        while(angle<end):
            x = R*mth.cos(angle)+X
            y = R*mth.sin(angle)+Y
            img[int(y)][int(x)]=BORDERCOLOR
            img[int(y)+1][int(x)]=BORDERCOLOR
            img[int(y)][int(x)+1]=BORDERCOLOR
            img[int(y)+1][int(x)+1]=BORDERCOLOR
            img[int(y)][int(x)-1]=BORDERCOLOR
            img[int(y)+1][int(x)]=BORDERCOLOR
            angle+=0.01
            for p in range(0,int(R)):
                x = p*mth.cos(angle)+X
                y = p*mth.sin(angle)+Y
                img[int(y)][int(x)]=COLOR
                img[int(y)+1][int(x)]=COLOR
                img[int(y)][int(x)+1]=COLOR
                img[int(y)+1][int(x)+1]=COLOR

    else:
        while(angle<end):
            x = R*mth.cos(angle)+X
            y = R*mth.sin(angle)+Y
            if(np.array_equal(img[int(y)][int(x)],WHITE)):
                img[int(y)][int(x)]=BORDERCOLOR
            if(np.array_equal(img[int(y)+1][int(x)],WHITE)):
                img[int(y)+1][int(x)]=BORDERCOLOR
            if(np.array_equal(img[int(y)][int(x)+1],WHITE)):
                img[int(y)][int(x)+1]=BORDERCOLOR
            if(np.array_equal(img[int(y)+1][int(x)+1],WHITE)):
                img[int(y)+1][int(x)+1]=BORDERCOLOR
            if(np.array_equal(img[int(y)][int(x)-1],WHITE)):
                img[int(y)][int(x)-1]=BORDERCOLOR
            if(np.array_equal(img[int(y)+1][int(x)-1],WHITE)):
                img[int(y)+1][int(x)-1]=BORDERCOLOR
            angle+=0.01
            for p in range(0,int(R)):
                x = p*mth.cos(angle)+X
                y = p*mth.sin(angle)+Y
                if(np.array_equal(img[int(y)][int(x)],WHITE)):
                    img[int(y)][int(x)]=COLOR
                    img[int(y)+1][int(x)]=COLOR
                    img[int(y)][int(x)+1]=COLOR
                    img[int(y)+1][int(x)+1]=COLOR

def GetCircleYByX(R,X,Y,x):
    y1=Y+(R**2-(x-X)**2)**0.5
    y2=Y-(R**2-(x-X)**2)**0.5
    return[y2,y1]

def GetCirclePointByAngle(R,X,Y,angle):
    x = R*mth.cos(angle)+X
```

```
    y = R*mth.sin(angle)+Y
    return[x,y]

def
DrowElipse(img,a,b,X,Y,COLOR,BORDERCOLOR=BLACK,start=0,end=2*mth.pi,over
lay=True):
    angle=start
    if(overlay):
        while(angle<end):
            x = a*mth.cos(angle)+X
            y = b*mth.sin(angle)+Y
            img[int(y)][int(x)]=BORDERCOLOR
            img[int(y)+1][int(x)]=BORDERCOLOR
            img[int(y)][int(x)+1]=BORDERCOLOR
            img[int(y)+1][int(x)+1]=BORDERCOLOR
            angle+=0.01
        x1=-a
        x2=+a
        y1=-b
        y2=+b
        for x in range(int(x1)+1,int(x2)-1):
            for y in range(int(y1)+1,int(y2)-1):
                if(x*x/(a*a)+y*y/(b*b)<1):
                    img[int(y+Y)][int(x+X)]=COLOR
    else:
        while(angle<end):
            x = a*mth.cos(angle)+X
            y = b*mth.sin(angle)+Y
            if(np.array_equal(img[int(y)][int(x)],WHITE)):
                img[int(y)][int(x)]=BORDERCOLOR
            if(np.array_equal(img[int(y)+1][int(x)],WHITE)):
                img[int(y)+1][int(x)]=BORDERCOLOR
            if(np.array_equal(img[int(y)][int(x)+1],WHITE)):
                img[int(y)][int(x)+1]=BORDERCOLOR
            if(np.array_equal(img[int(y)+1][int(x)+1],WHITE)):
                img[int(y)+1][int(x)+1]=BORDERCOLOR
            angle+=0.01
        x1=-a
        x2=+a
        y1=-b
        y2=+b
        for  x  in    range(int(x1)+1,int(x2)-1):
            for y in range(int(y1)+1,int(y2)-1):
                if(np.array_equal(img[int(y+Y)][int(x+X)],WHITE)):
                    if(x*x/(a*a)+y*y/(b*b)<1):
                        img[int(y+Y)][int(x+X)]=COLOR

def
DrowElipseConture(img,a,b,X,Y,BORDERCOLOR=BLACK,start=0,end=2*mth.pi,ove
rlay=True):
    angle=start
    if(overlay):
        while(angle<end):
            x = a*mth.cos(angle)+X
            y = b*mth.sin(angle)+Y
            img[int(y)][int(x)]=BORDERCOLOR
            img[int(y)+1][int(x)]=BORDERCOLOR
```

```
                img[int(y)][int(x)+1]=BORDERCOLOR
                img[int(y)+1][int(x)+1]=BORDERCOLOR
                img[int(y)][int(x)-1]=BORDERCOLOR
                img[int(y)+1][int(x)]=BORDERCOLOR
                angle+=0.01
        else:
            while(angle<end):
                x = a*mth.cos(angle)+X
                y = b*mth.sin(angle)+Y
                if(np.array_equal(img[int(y)][int(x)],WHITE)):
                    img[int(y)][int(x)]=BORDERCOLOR
                if(np.array_equal(img[int(y)+1][int(x)],WHITE)):
                    img[int(y)+1][int(x)]=BORDERCOLOR
                if(np.array_equal(img[int(y)][int(x)+1],WHITE)):
                    img[int(y)][int(x)+1]=BORDERCOLOR
                if(np.array_equal(img[int(y)+1][int(x)+1],WHITE)):
                    img[int(y)+1][int(x)+1]=BORDERCOLOR
                if(np.array_equal(img[int(y)][int(x)-1],WHITE)):
                    img[int(y)][int(x)-1]=BORDERCOLOR
                if(np.array_equal(img[int(y)+1][int(x)-1],WHITE)):
                    img[int(y)+1][int(x)-1]=BORDERCOLOR
                angle+=0.01


    def GetElipsePointByAngle(a,b,X,Y,angle):
        x = a*mth.cos(angle)+X
        y = b*mth.sin(angle)+Y
        return[x,y]

    def
    DrowParabola(img,points,borders,COLOR,BORDERCOLOR=BLACK,part=True,overla
    y=True):
        x1=points[0][0]
        y1=points[0][1]
        x2=points[1][0]
        y2=points[1][1]
        x3=points[2][0]
        y3=points[2][1]

        XB=borders[0][0]
        YB=borders[0][1]
        XE=borders[1][0]
        YE=borders[1][1]

        matrix=np.zeros([3,4])
        matrix[0][0]=x1*x1
        matrix[0][1]=x1
        matrix[0][2]=1
        matrix[0][3]=y1

        matrix[1][0]=x2*x2
        matrix[1][1]=x2
        matrix[1][2]=1
        matrix[1][3]=y2

        matrix[2][0]=x3*x3
        matrix[2][1]=x3
```

```
    matrix[2][2]=1
    matrix[2][3]=y3
    for i in range(3):
        a = matrix[i][i]
        for j in range(4):
            matrix[i][j]/=a
        for k in range(3):
            a=matrix[k][i]
            for j in range(i,4):
                if(k!=i):
                    matrix[k][j]-=a*matrix[i][j]
    a = matrix[0][3]
    b = matrix[1][3]
    c = matrix[2][3]


    if(overlay):
        for x in range(int(XB),int(XE)+1):
            y = a*x*x+b*x+c
            img[int(y)][x]=BORDERCOLOR
            img[int(y)+1][x]=BORDERCOLOR
            img[int(y)-1][x]=BORDERCOLOR
            if(part==False):
                for i in range(int(y)+1,int(YE)-1):
                    img[int(i)][x]=COLOR
            else:
                for i in range(int(YB)+1,int(y)-1):
                    if(np.array_equal(img[int(i)][x],BORDERCOLOR)==False
):
                        img[int(i)][x]=COLOR
    else:
        for x in range(int(XB),int(XE)+1):
            y = a*x*x+b*x+c
            if(np.array_equal(img[int(y)][int(x)],WHITE)):
                img[int(y)][x]=BORDERCOLOR
            if(np.array_equal(img[int(y)+1][int(x)],WHITE)):
                img[int(y)+1][x]=BORDERCOLOR
            if(np.array_equal(img[int(y)-1][int(x)],WHITE)):
                img[int(y)-1][x]=BORDERCOLOR
            if(part==False):
                for i in range(int(y)+1,int(YE)-1):
                    if(np.array_equal(img[int(i)][x],WHITE)):
                        img[int(i)][x]=COLOR
            else:
                for i in range(int(YB)+1,int(y)-1):
                    if(np.array_equal(img[int(i)][x],WHITE)):
                        img[int(i)][x]=COLOR

    return a,b,c


def
DrowHiperbolaByPoints(img,width,X,Y,point,YB,YE,COLOR,BORDERCOLOR=BLACK,
overlay=True):
    x1=-width/2
    y1=0
    x2=point[0]-X
```

```
    y2=point[1]-Y
    b2 = (y1*y1*x2*x2-y2*y2*x1*x1)/(x1*x1-x2*x2)
    a2 = (x1*x1)/(1+(y1*y1)/b2)
    if(overlay):
        for y in range(int(YB-Y),int(YE-Y)):
            x1 = X+(a2+(a2*y*y)/b2)**0.5
            x2 = X-(a2+(a2*y*y)/b2)**0.5
            img[int(y+Y)][int(x1)]=BORDERCOLOR
            img[int(y+Y)][int(x1)+1]=BORDERCOLOR
            img[int(y+Y)][int(x2)]=BORDERCOLOR
            img[int(y+Y)][int(x2)+1]=BORDERCOLOR
            img[int(y+Y+1)][int(x1)]=BORDERCOLOR
            img[int(y+Y+1)][int(x1)+1]=BORDERCOLOR
            img[int(y+Y+1)][int(x2)]=BORDERCOLOR
            img[int(y+Y+1)][int(x2)+1]=BORDERCOLOR
            for x in range(int(x2),int(x1)+1):
                img[int(y+Y)][x]=COLOR
    else:
        for y in range(int(YB-Y),int(YE-Y)):
            x1 = X+(a2+(a2*y*y)/b2)**0.5
            x2 = X-(a2+(a2*y*y)/b2)**0.5
            if(np.array_equal(img[int(y+Y)][int(x1)],WHITE)):
                img[int(y+Y)][int(x1)]=BORDERCOLOR
            if(np.array_equal(img[int(y+Y)][int(x1)+1],WHITE)):
                img[int(y+Y)][int(x1)+1]=BORDERCOLOR
            if(np.array_equal(img[int(y+Y)][int(x2)],WHITE)):
                img[int(y+Y)][int(x2)]=BORDERCOLOR
            if(np.array_equal(img[int(y+Y)][int(x2)+1],WHITE)):
                img[int(y+Y)][int(x2)+1]=BORDERCOLOR
            if(np.array_equal(img[int(y+Y)+1][int(x1)],WHITE)):
                img[int(y+Y+1)][int(x1)]=BORDERCOLOR
            if(np.array_equal(img[int(y+Y)+1][int(x1)+1],WHITE)):
                img[int(y+Y+1)][int(x1)+1]=BORDERCOLOR
            if(np.array_equal(img[int(y+Y)+1][int(x2)],WHITE)):
                img[int(y+Y+1)][int(x2)]=BORDERCOLOR
            if(np.array_equal(img[int(y+Y)+1][int(x2)+1],WHITE)):
                img[int(y+Y+1)][int(x2)+1]=BORDERCOLOR
            for x in range(int(x2),int(x1)+1):
                if(np.array_equal(img[int(y+Y)][x],WHITE)):
                    img[int(y+Y)][x]=COLOR
    return a2,b2


def
DrowHiperbolaByBParam(img,width,X,Y,b2,YB,YE,COLOR,BORDERCOLOR=BLACK,ove
rlay=True):
    x1=-width/2
    y1=0
    a2 = (x1*x1)/(1+(y1*y1)/b2)
    if(overlay):
        for y in range(YB-Y,YE-Y):
            x1 = X+(a2+(a2*y*y)/b2)**0.5
            x2 = X-(a2+(a2*y*y)/b2)**0.5
            img[int(y+Y)][int(x1)]=BORDERCOLOR
            img[int(y+Y)][int(x1)+1]=BORDERCOLOR
            img[int(y+Y)][int(x2)]=BORDERCOLOR
            img[int(y+Y)][int(x2)+1]=BORDERCOLOR
```

```
            img[int(y+Y+1)][int(x1)]=BORDERCOLOR
            img[int(y+Y+1)][int(x1)+1]=BORDERCOLOR
            img[int(y+Y+1)][int(x2)]=BORDERCOLOR
            img[int(y+Y+1)][int(x2)+1]=BORDERCOLOR
            for x in range(int(x2),int(x1)+1):
                img[int(y+Y)][x]=COLOR
    else:
        for y in range(YB-Y,YE-Y):
            x1 = X+(a2+(a2*y*y)/b2)**0.5
            x2 = X-(a2+(a2*y*y)/b2)**0.5
            if(np.array_equal(img[int(y+Y)][int(x1)],WHITE)):
                img[int(y+Y)][int(x1)]=BORDERCOLOR
            if(np.array_equal(img[int(y+Y)][int(x1)+1],WHITE)):
                img[int(y+Y)][int(x1)+1]=BORDERCOLOR
            if(np.array_equal(img[int(y+Y)][int(x2)],WHITE)):
                img[int(y+Y)][int(x2)]=BORDERCOLOR
            if(np.array_equal(img[int(y+Y)][int(x2)+1],WHITE)):
                img[int(y+Y)][int(x2)+1]=BORDERCOLOR
            if(np.array_equal(img[int(y+Y)+1][int(x1)],WHITE)):
                img[int(y+Y+1)][int(x1)]=BORDERCOLOR
            if(np.array_equal(img[int(y+Y)+1][int(x1)+1],WHITE)):
                img[int(y+Y+1)][int(x1)+1]=BORDERCOLOR
            if(np.array_equal(img[int(y+Y)+1][int(x2)],WHITE)):
                img[int(y+Y+1)][int(x2)]=BORDERCOLOR
            if(np.array_equal(img[int(y+Y)+1][int(x2)+1],WHITE)):
                img[int(y+Y+1)][int(x2)+1]=BORDERCOLOR
            for x in range(int(x2),int(x1)+1):
                if(np.array_equal(img[int(y+Y)][x],WHITE)):
                    img[int(y+Y)][x]=COLOR
    return a2,b2


def
DrowHiperbola(img,point,X,Y,b2,YB,YE,COLOR,BORDERCOLOR=BLACK,overlay=Tru
e):
    x1=point[0]-X
    y1=point[1]-Y
    a2 = (x1*x1)/(1+(y1*y1)/b2)

    if(overlay):
        for y in range(int(YB-Y),int(YE-Y)):
            x1 = X+(a2+(a2*y*y)/b2)**0.5
            x2 = X-(a2+(a2*y*y)/b2)**0.5
            img[int(y+Y)][int(x1)]=BORDERCOLOR
            img[int(y+Y)][int(x1)+1]=BORDERCOLOR
            img[int(y+Y)][int(x2)]=BORDERCOLOR
            img[int(y+Y)][int(x2)+1]=BORDERCOLOR
            img[int(y+Y+1)][int(x1)]=BORDERCOLOR
            img[int(y+Y+1)][int(x1)+1]=BORDERCOLOR
            img[int(y+Y+1)][int(x2)]=BORDERCOLOR
            img[int(y+Y+1)][int(x2)+1]=BORDERCOLOR
            for x in range(int(x2),int(x1)+1):
                    img[int(y+Y)][x]=COLOR
    else:
        for y in range(int(YB-Y),int(YE-Y)):
            x1 = X+(a2+(a2*y*y)/b2)**0.5
            x2 = X-(a2+(a2*y*y)/b2)**0.5
```

```
            if(np.array_equal(img[int(y+Y)][int(x1)],WHITE)):
                img[int(y+Y)][int(x1)]=BORDERCOLOR
            if(np.array_equal(img[int(y+Y)][int(x1)+1],WHITE)):
                img[int(y+Y)][int(x1)+1]=BORDERCOLOR
            if(np.array_equal(img[int(y+Y)][int(x2)],WHITE)):
                img[int(y+Y)][int(x2)]=BORDERCOLOR
            if(np.array_equal(img[int(y+Y)][int(x2)+1],WHITE)):
                img[int(y+Y)][int(x2)+1]=BORDERCOLOR
            if(np.array_equal(img[int(y+Y)+1][int(x1)],WHITE)):
                img[int(y+Y+1)][int(x1)]=BORDERCOLOR
            if(np.array_equal(img[int(y+Y)+1][int(x1)+1],WHITE)):
                img[int(y+Y+1)][int(x1)+1]=BORDERCOLOR
            if(np.array_equal(img[int(y+Y)+1][int(x2)],WHITE)):
                img[int(y+Y+1)][int(x2)]=BORDERCOLOR
            if(np.array_equal(img[int(y+Y)+1][int(x2)+1],WHITE)):
                img[int(y+Y+1)][int(x2)+1]=BORDERCOLOR
            for x in range(int(x2),int(x1)+1):
                if(np.array_equal(img[int(y+Y)][x],WHITE)):
                    img[int(y+Y)][x]=COLOR
    return a2,b2

def GetHiperbolaPointByBParam(width,X,Y,b2,Y1):
    x1=-width/2
    y1=0
    a2 = (x1*x1)/(1+(y1*y1)/b2)
    y=Y1-Y
    x1 = X+(a2+(a2*y*y)/b2)**0.5
    x2 = X-(a2+(a2*y*y)/b2)**0.5
    return [x1,x2]


def
DrowBrunch(img,points,COLOR,BORDERCOLOR=BLACK,part=True,overlay=True):
    if(points[0][1]>points[1][1]):
        GX = points[1][0]
        GY = points[1][1]
    else:
        GX = points[0][0]
        GY = points[0][1]
    x1 = points[0][0]-GX
    y1 = points[0][1]-GY
    x2 = points[1][0]-GX
    y2 = points[1][1]-GY

    if(y1>y2):
        p2 = y1*y1/x1
    else:
        p2 = y2*y2/x2

    if(overlay):
        for x in range(int(x1),int(x2)+1):
            y = GY + (p2*x)**0.5
            img[int(y)][int(x+GX)]=BORDERCOLOR
            img[int(y)+1][int(x+GX)]=BORDERCOLOR
            img[int(y)][int(x+GX)+1]=BORDERCOLOR
            img[int(y)+1][int(x+GX)+1]=BORDERCOLOR
            if(y1>y2):
```

```
                if(part):
                    for i in range(int(y),int(y1+GY)):
                        img[int(i)][int(x+GX)]=COLOR
                else:
                    for i in range(int(y2+GY),int(y)):
                        img[int(i)][int(x+GX)]=COLOR
            else:
                if(part):
                    for i in range(int(y),int(y2+GY)):
                        img[int(i)][int(x+GX)]=COLOR
                else:
                    for i in range(int(y1+GY),int(y)):
                        img[int(i)][int(x+GX)]=COLOR
    else:
        for x in range(int(x1),int(x2)+1):
            y = GY + (p2*x)**0.5
            if(np.array_equal(img[int(y)][int(x+GX)],WHITE)):
                img[int(y)][int(x+GX)]=BORDERCOLOR
            if(np.array_equal(img[int(y)+1][int(x+GX)],WHITE)):
                img[int(y)+1][int(x+GX)]=BORDERCOLOR
            if(np.array_equal(img[int(y)][int(x+GX)],WHITE)):
                img[int(y)][int(x+GX)+1]=BORDERCOLOR
            if(np.array_equal(img[int(y)+1][int(x+GX)],WHITE)):
                img[int(y)+1][int(x+GX)+1]=BORDERCOLOR
            if(y1>y2):
                if(part):
                    for i in range(int(y),int(y1+GY)):
                        if(np.array_equal(img[int(i)][int(x+GX)],WHITE))
:
                            img[int(i)][int(x+GX)]=COLOR
                else:
                    for i in range(int(y2+GY),int(y)):
                        if(np.array_equal(img[int(i)][int(x+GX)],WHITE))
:
                            img[int(i)][int(x+GX)]=COLOR
            else:
                if(part):
                    for i in range(int(y),int(y2+GY)):
                        if(np.array_equal(img[int(i)][int(x+GX)],WHITE))
:
                            img[int(i)][int(x+GX)]=COLOR
                else:
                    for i in range(int(y1+GY),int(y)):
                        if(np.array_equal(img[int(i)][int(x+GX)],WHITE))
:
                            img[int(i)][int(x+GX)]=COLOR
    return p2

def GetBrunchPointByY(p2,y,GX,GY):
    return ((y - GY)**2)/p2+GX

def DrowDoublePolynom(img,
points,X,COLOR,BORDERCOLOR=BLACK,overlay=True):
    x1=points[0][0]
    y1=points[0][1]
    x2=points[1][0]
    y2=points[1][1]
```

```
x3=points[2][0]
y3=points[2][1]
x4=points[3][0]
y4=points[3][1]

matrix=np.zeros([4,5])
matrix[0][0]=y1*y1*y1
matrix[0][1]=y1*y1
matrix[0][2]=y1
matrix[0][3]=1
matrix[0][4]=x1

matrix[1][0]=y2*y2*y2
matrix[1][1]=y2*y2
matrix[1][2]=y2
matrix[1][3]=1
matrix[1][4]=x2

matrix[2][0]=y3*y3*y3
matrix[2][1]=y3*y3
matrix[2][2]=y3
matrix[2][3]=1
matrix[2][4]=x3

matrix[3][0]=y4*y4*y4
matrix[3][1]=y4*y4
matrix[3][2]=y4
matrix[3][3]=1
matrix[3][4]=x4
for i in range(4):
    a = matrix[i][i]
    for j in range(5):
        matrix[i][j]/=a
    for k in range(4):
        a=matrix[k][i]
        for j in range(i,5):
            if(k!=i):
                matrix[k][j]-=a*matrix[i][j]
a = matrix[0][4]
b = matrix[1][4]
c = matrix[2][4]
d = matrix[3][4]
beg = min(y1,y2,y3,y4)
e = max(y1,y2,y3,y4)

if(overlay):
    for y in range(int(beg),int(e)+1):
        x = a*y*y*y+b*y*y+c*y+d
        mx = int(X*2-x)
        img[int(y)][int(x)]=BORDERCOLOR
        img[int(y)+1][int(x)]=BORDERCOLOR
        img[int(y)-1][int(x)]=BORDERCOLOR

        img[int(y)][int(mx)]=BORDERCOLOR
        img[int(y)+1][int(mx)]=BORDERCOLOR
        img[int(y)-1][int(mx)]=BORDERCOLOR
        begin=int(min(x,mx))
```

```
            end=int(max(x,mx))

            for i in range(begin,end):
                img[int(y)][i]=COLOR
                img[int(y)+1][i]=COLOR
                img[int(y)-1][i]=COLOR
    else:
        for y in range(int(beg),int(e)+1):
            x = a*y*y*y+b*y*y+c*y+d
            mx = int(X*2-x)
            if(np.array_equal(img[int(y)][int(x)],WHITE)):
                img[int(y)][int(x)]=BORDERCOLOR
            if(np.array_equal(img[int(y)+1][int(x)],WHITE)):
                img[int(y)+1][int(x)]=BORDERCOLOR
            if(np.array_equal(img[int(y)-1][int(x)],WHITE)):
                img[int(y)-1][int(x)]=BORDERCOLOR

            if(np.array_equal(img[int(y)][int(mx)],WHITE)):
                img[int(y)][int(mx)]=BORDERCOLOR
            if(np.array_equal(img[int(y)+1][int(mx)],WHITE)):
                img[int(y)+1][int(mx)]=BORDERCOLOR
            if(np.array_equal(img[int(y)-1][int(mx)],WHITE)):
                img[int(y)-1][int(mx)]=BORDERCOLOR
            begin=int(min(x,mx))
            end=int(max(x,mx))

            for i in range(begin,end):
                if(np.array_equal(img[int(y)][i],WHITE)):
                    img[int(y)][i]=COLOR
                if(np.array_equal(img[int(y)+1][i],WHITE)):
                    img[int(y)+1][i]=COLOR
                if(np.array_equal(img[int(y)-1][i],WHITE)):
                    img[int(y)-1][i]=COLOR
    return [a,b,c,d]


def
DrowDoublePolynomByParams(img,params,X,COLOR,borders,BORDERCOLOR=BLACK,o
verlay=True):
    a = params[0]
    b = params[1]
    c = params[2]
    d = params[3]
    beg = borders[0]
    e = borders[1]
    if(overlay):
        for y in range(int(beg),int(e)+1):
            x = a*y*y*y+b*y*y+c*y+d
            mx = int(X*2-x)
            img[int(y)][int(x)]=BORDERCOLOR
            img[int(y)+1][int(x)]=BORDERCOLOR
            img[int(y)-1][int(x)]=BORDERCOLOR

            img[int(y)][int(mx)]=BORDERCOLOR
            img[int(y)+1][int(mx)]=BORDERCOLOR
            img[int(y)-1][int(mx)]=BORDERCOLOR
```

```
                begin=int(min(x,mx))
                end=int(max(x,mx))

                for i in range(begin,end):
                    if(np.array_equal(img[int(y)][i],BORDERCOLOR)==False):
                        img[int(y)][i]=COLOR
                    if(np.array_equal(img[int(y)+1][i],BORDERCOLOR)==False):
                        img[int(y)+1][i]=COLOR
                    if(np.array_equal(img[int(y)-1][i],BORDERCOLOR)==False):
                        img[int(y)-1][i]=COLOR
        else:
            for y in range(int(beg),int(e)+1):
                x = a*y*y*y+b*y*y+c*y+d
                mx = int(X*2-x)
                if(np.array_equal(img[int(y)][int(x)],WHITE)):
                    img[int(y)][int(x)]=BORDERCOLOR
                if(np.array_equal(img[int(y)+1][int(x)],WHITE)):
                    img[int(y)+1][int(x)]=BORDERCOLOR
                if(np.array_equal(img[int(y)-1][int(x)],WHITE)):
                    img[int(y)-1][int(x)]=BORDERCOLOR

                if(np.array_equal(img[int(y)][int(mx)],WHITE)):
                    img[int(y)][int(mx)]=BORDERCOLOR
                if(np.array_equal(img[int(y)+1][int(mx)],WHITE)):
                    img[int(y)+1][int(mx)]=BORDERCOLOR
                if(np.array_equal(img[int(y)-1][int(mx)],WHITE)):
                    img[int(y)-1][int(mx)]=BORDERCOLOR
                begin=int(min(x,mx))
                end=int(max(x,mx))

                for i in range(begin,end):
                    if(np.array_equal(img[int(y)][i],WHITE)):
                        img[int(y)][i]=COLOR
                    if(np.array_equal(img[int(y)+1][i],WHITE)):
                        img[int(y)+1][i]=COLOR
                    if(np.array_equal(img[int(y)-1][i],WHITE)):
                        img[int(y)-1][i]=COLOR
        return [a,b,c,d]


    def GetPolynomXByyandParam(y,params):
        a = params[0]
        b = params[1]
        c = params[2]
        d = params[3]
        return a*y*y*y+b*y*y+c*y+d

    def DrowRectangle(img,centr,halfA,halfB,COLOR,overlay=True):
        if(overlay):
            for x in range(int(centr[0]-halfA)-1,int(centr[0]+halfA)+1):
                for y in range(int(centr[1]-halfB)-1,int(centr[1]+halfB)+1):
                    img[y][x]=COLOR
        else:
            for x in range(int(centr[0]-halfA)-1,int(centr[0]+halfA)+1):
                for y in range(int(centr[1]-halfB)-1,int(centr[1]+halfB)+1):
                    if(np.array_equal(img[y][x],WHITE)):
```

```
                    img[y][x]=COLOR

def DrowLine(img,point1,point2,BORDERCOLOR=BLACK,overlay=True):
    x1=point1[0]
    y1=point1[1]
    x2=point2[0]
    y2=point2[1]

    k=(y2-y1)/(x2-x1)
    if(overlay):
        if(mth.fabs(k)<=1):
            b = y1-(x1*(y2-y1))/(x2-x1)
            beg = min(x1,x2)
            end = max(x1,x2)
            for x in range(int(beg),int(end)):
                y=k*x+b
                img[int(y)][x]=BORDERCOLOR
                img[int(y)+1][x]=BORDERCOLOR
                img[int(y)-1][x]=BORDERCOLOR
        else:
            k = (x2-x1)/(y2-y1)
            b = x1-(y1*(x2-x1))/(y2-y1)
            beg = min(y1,y2)
            end = max(y1,y2)
            for y in range(int(beg),int(end)):
                x=k*y+b
                img[y][int(x)]=BORDERCOLOR
                img[y+1][int(x)]=BORDERCOLOR
                img[y-1][int(x)]=BORDERCOLOR
    else:
        if(mth.fabs(k)<=1):
            b = y1-(x1*(y2-y1))/(x2-x1)
            beg = min(x1,x2)
            end = max(x1,x2)
            for x in range(int(beg),int(end)):
                y=k*x+b
                if(np.array_equal(img[int(y)][x],WHITE)):
                    img[int(y)][x]=BORDERCOLOR
                if(np.array_equal(img[int(y)+1][x],WHITE)):
                    img[int(y)+1][x]=BORDERCOLOR
                if(np.array_equal(img[int(y)-1][x],WHITE)):
                    img[int(y)-1][x]=BORDERCOLOR
        else:
            k = (x2-x1)/(y2-y1)
            b = x1-(y1*(x2-x1))/(y2-y1)
            beg = min(y1,y2)
            end = max(y1,y2)
            for y in range(int(beg),int(end)):
                x=k*y+b
                if(np.array_equal(img[y][int(x)],WHITE)):
                    img[y][int(x)]=BORDERCOLOR
                if(np.array_equal(img[y+1][int(x)],WHITE)):
                    img[y+1][int(x)]=BORDERCOLOR
                if(np.array_equal(img[y-1][int(x)],WHITE)):
                    img[y-1][int(x)]=BORDERCOLOR
```

```
def DrowColorLine(img,point1,point2,COLOR,part =
True,BORDERCOLOR=BLACK,overlay=True):
    x1=point1[0]
    y1=point1[1]
    x2=point2[0]
    y2=point2[1]

    k=(y2-y1)/(x2-x1)
    if(overlay):
        if(mth.fabs(k)<=1):
            b = y1-(x1*(y2-y1))/(x2-x1)
            beg = min(x1,x2)
            end = max(x1,x2)
            for x in range(int(beg),int(end)):
                y=k*x+b
                img[int(y)][x]=BORDERCOLOR
                img[int(y)+1][x]=BORDERCOLOR
                img[int(y)-1][x]=BORDERCOLOR
                img[int(y)][x+1]=BORDERCOLOR
                img[int(y)+1][x+1]=BORDERCOLOR
                img[int(y)-1][x+1]=BORDERCOLOR
        else:
            k = (x2-x1)/(y2-y1)
            b = x1-(y1*(x2-x1))/(y2-y1)
            beg = min(y1,y2)
            end = max(y1,y2)
            for y in range(int(beg),int(end)):
                x=k*y+b
                img[y][int(x)]=BORDERCOLOR
                img[y+1][int(x)]=BORDERCOLOR
                img[y-1][int(x)]=BORDERCOLOR
                if(part):
                    for i in range(int(min(x1,x2)),int(x)):
                        if(np.array_equal(img[y][i],BORDERCOLOR)==False)
:
                            img[y][i]=COLOR
                else:
                    for i in range(int(x),int(max(x1,x2))):
                        if(np.array_equal(img[y][i],BORDERCOLOR)==False)
:
                            img[y][i]=COLOR
    else:
        if(mth.fabs(k)<=1):
            b = y1-(x1*(y2-y1))/(x2-x1)
            beg = min(x1,x2)
            end = max(x1,x2)
            for x in range(int(beg),int(end)):
                y=k*x+b
                if(np.array_equal(img[int(y)][x],WHITE)):
                    img[int(y)][x]=BORDERCOLOR
                if(np.array_equal(img[int(y)+1][x],WHITE)):
                    img[int(y)+1][x]=BORDERCOLOR
                if(np.array_equal(img[int(y)-1][x],WHITE)):
                    img[int(y)-1][x]=BORDERCOLOR
                if(np.array_equal(img[int(y)][x+1],WHITE)):
                    img[int(y)][x+1]=BORDERCOLOR
                if(np.array_equal(img[int(y)+1][x+1],WHITE)):
```

```
                    img[int(y)+1][x+1]=BORDERCOLOR
                if(np.array_equal(img[int(y)-1][x+1],WHITE)):
                    img[int(y)-1][x+1]=BORDERCOLOR
                if(part):
                    for i in range(int(min(x1,x2)),int(x)):
                        if(np.array_equal(img[y][i],WHITE)):
                            img[y][i]=COLOR
                else:
                    for i in range(int(x),int(max(x1,x2))):
                        if(np.array_equal(img[y][i],WHITE)):
                            img[y][i]=COLOR
        else:
            k = (x2-x1)/(y2-y1)
            b = x1-(y1*(x2-x1))/(y2-y1)
            beg = min(y1,y2)
            end = max(y1,y2)
            for y in range(int(beg),int(end)):
                x=k*y+b
                if(np.array_equal(img[y][int(x)],WHITE)):
                    img[y][int(x)]=BORDERCOLOR
                if(np.array_equal(img[y+1][int(x)],WHITE)):
                    img[y+1][int(x)]=BORDERCOLOR
                if(np.array_equal(img[y-1][int(x)],WHITE)):
                    img[y-1][int(x)]=BORDERCOLOR

def Drow2Polynom(img,
points,COLOR,BORDERCOLOR=BLACK,part=True,overlay=True):
    x1=points[0][0]
    y1=points[0][1]
    x2=points[1][0]
    y2=points[1][1]
    x3=points[2][0]
    y3=points[2][1]

    matrix=np.zeros([3,4])
    matrix[0][0]=y1*y1
    matrix[0][1]=y1
    matrix[0][2]=1
    matrix[0][3]=x1

    matrix[1][0]=y2*y2
    matrix[1][1]=y2
    matrix[1][2]=1
    matrix[1][3]=x2

    matrix[2][0]=y3*y3
    matrix[2][1]=y3
    matrix[2][2]=1
    matrix[2][3]=x3

    for i in range(3):
        a = matrix[i][i]
        for j in range(4):
            matrix[i][j]/=a
        for k in range(3):
            a=matrix[k][i]
            for j in range(i,4):
```

```
                if(k!=i):
                    matrix[k][j]-=a*matrix[i][j]
    a = matrix[0][3]
    b = matrix[1][3]
    c = matrix[2][3]
    beg = min(y1,y2,y3)
    e = max(y1,y2,y3)

    if(overlay):
        for y in range(int(beg),int(e)+1):
            x = a*y*y+b*y+c
            img[int(y)][int(x)]=BORDERCOLOR
            img[int(y)+1][int(x)]=BORDERCOLOR
            img[int(y)-1][int(x)]=BORDERCOLOR
            if(part):
                begin=int(min(x1,x2,x3))
                end=int(x)
            else:
                begin=int(x)+1
                end = int(max(x1,x2,x3))
            for i in range(begin,end):
                if(np.array_equal(img[int(y)][i],BORDERCOLOR)==False):
                    img[int(y)][i]=COLOR
                if(np.array_equal(img[int(y)+1][i],BORDERCOLOR)==False):
                    img[int(y)+1][i]=COLOR
                if(np.array_equal(img[int(y)-1][i],BORDERCOLOR)==False):
                    img[int(y)-1][i]=COLOR
    else:
        for y in range(int(beg),int(e)+1):
            x = a*y*y+b*y+c
            if(np.array_equal(img[int(y)][int(x)],WHITE)):
                img[int(y)][int(x)]=BORDERCOLOR
            if(np.array_equal(img[int(y)+1][int(x)],WHITE)):
                img[int(y)+1][int(x)]=BORDERCOLOR
            if(np.array_equal(img[int(y)-1][int(x)],WHITE)):
                img[int(y)-1][int(x)]=BORDERCOLOR

            if(part):
                begin=int(min(x1,x2,x3))
                end=int(x)
            else:
                begin=int(x)
                end = int(max(x1,x2,x3))

            for i in range(begin,end):
                if(np.array_equal(img[int(y)][i],WHITE)):
                    img[int(y)][i]=COLOR
                if(np.array_equal(img[int(y)+1][i],WHITE)):
                    img[int(y)+1][i]=COLOR
                if(np.array_equal(img[int(y)-1][i],WHITE)):
                    img[int(y)-1][i]=COLOR
    return [a,b,c]
```