МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ТЕХНОЛОГІЙ ТА ДИЗАЙНУ

ФАКУЛЬТЕТ МЕХАТРОНІКИ ТА КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ

КАФЕДРА КОМП'ЮТЕРНИХ НАУК

# КВАЛІФІКАЦІЙНА РОБОТА

на тему:

Розроблення математичного та програмного забезпечення для автоматизованої підготовки інформації про деталі шкіргалантереї

Рівень вищої освіти другий (магістерський)

Спеціальність 122 Комп'ютерні науки

Освітня програма Комп'ютерні науки

Виконав: студент групи МгДІТ-22
Марія КАЗЕМИРЧИК
Науковий керівник: к.т.н. Наталія ЧУПРИНКА
Рецензент д.т.н., проф. Сергій КРАСНИТСЬКИЙ

Київ 2023

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ТЕХНОЛОГІЙ ТА ДИЗАЙНУ

Факультет мехатроніки та комп'ютерних технологій

Кафедра комп'ютерні науки

Рівень вищої освіти другий (магістерський)

Спеціальність 122 Комп'ютерні науки

Освітня програма Комп'ютерні науки

**ЗАТВЕРДЖУЮ**

Завідувач кафедри КН

_____Володимир ЩЕРБАНЬ

«___»_____2023__року

**З А В Д А Н Н Я**

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТА

**Каземирчик Марії Сергіївни**

1. Тема роботи: Розроблення математичного та програмного забезпечення для автоматизованої підготовки інформації про деталі шкіргалантереї, науковий керівник роботи Чупринка Н. В., к.т.н., доцент, затверджені наказом вищого навчального закладу від "12" вересня 2023 року № 210-уч.
2. Строк подання студентом роботи: 08.11.2023р.
3. Вихідні дані до роботи: Математичні моделі та алгоритми, розроблені для автоматизованої підготовки інформації про деталі шкіргалантереї; Літературні огляди цієї теми.
4. Зміст дипломної роботи:
РОЗДІЛ 1 Постановка проблеми автоматизованої підготовки інформації про деталі шкіргалантереї;
РОЗДІЛ 2 Теоретичний огляд;
РОЗДІЛ 3 Розробка та дослідження.
5. Перелік графічного матеріалу: 12 рис.
6. Дата видачі завдання: 09.2023 р. .

# КАЛЕНДАРНИЙ ПЛАН

| № з/п | Назва етапів дипломної магістерської роботи | Терміни виконання етапів |
|---|---|---|
| 1 | Вступ | 01.10.2023 |
| 2 | Розділ 1: Постановка проблеми автоматизованої підготовки інформації про деталі шкіргалантереї | 02.10.2023 |
| 3 | Розділ 2: Теоретичний огляд | 10.10.2023 |
| 4 | Розділ 3: Розробка та дослідження | 20.10.2023 |
| 5 | Висновок | 30.10.2023 |
| 6 | Оформлення дипломної магістерської роботи (чистовий варіант) | 01.11.2023 |
| 7 | Здача дипломної магістерської роботи на кафедру для рецензування (за 14 днів до захисту) | 01.11.2023 |
| 8 | Перевірка дипломної магістерської роботи на наявність ознак плагіату (за 10 днів до захисту) | 08.11.2023 |
| 9 | Подання дипломної магістерської роботи у відділ магістратури для перевірки виконання додатку до індивідуального навчального плану (за 10 днів до захисту) | 08.11.2023 |
| 10 | Подання дипломної магістерської роботи на затвердження завідувачу кафедри (з 7 днів до захисту) | 08.11.2023 |

З завданням ознайомлений:

Студент: _____ Марія КАЗЕМИРЧИК

Науковий керівник роботи _____ Наталія ЧУПРИНКА

# АНОТАЦІЯ

**Каземирчик Марія. Тема: Розроблення математичного та програмного забезпечення для автоматизованої підготовки інформації про деталі шкіргалантереї.**

Дипломна магістерська робота за спеціальністю 122 - «Комп'ютерні науки». – Київський національний університет технологій та дизайну, Київ, 2023 рік.

Мета роботи – розробити та випробувати програму для ефективної обробки даних шкіргалантереї з подальшим можливим використанням у медичних дослідженнях. Для реалізації поставленою мети поставлені наступні задачі:

- Розробка програмного забезпечення.
- Автоматизація обробки даних.
- Тестування та аналіз результатів.

Загальний обсяг роботи: 42 сторінок, 12 рисунків, 17 посилань, 3 таблиці.

*Ключові слова:* HTML, Java, автоматизація виробництва шкіргалантереї.

# ANNOTATION

**Kazemyrchyk Mariia. Development of Mathematical and Software Solutions for Automated Data Preparation in Skincare.**

Master's Thesis in Computer Science. – Kyiv National University of Technologies and Design, Kyiv, 2023.

The aim of this work is to develop and test software for efficient skincare data processing with potential use in medical research.

To achieve this goal, the following tasks are set:

- Software development.
- Data processing automation.
- Testing and results analysis.

Total work volume: 42 pages, 12 figures, 17 reference.

Keywords: HTML, Java.

# ЗМІСТ

# ВСТУП

Сучасні технології вимагають ефективного та точного збору, обробки та підготовки інформації про деталі шкіргалантереї. Це стає важливим завданням у контексті постійного розвитку легкої галузі та високої конкуренції на ринку цих товарів.

*Метою* даної магістерської роботи є розробка математичного та програмного забезпечення для автоматизованої підготовки інформації про деталі шкіргалантереї. Ця робота спрямована на вирішення актуальної проблеми у галузі шкіргалантереї, а саме на ускладнення та оптимізацію процесу підготовки інформації про деталі шкіргалантереї.

Основні *завдання* роботи включають:

1. Визначення важливості та актуальності проблеми автоматизованої підготовки інформації про деталі шкіргалантереї в контексті сучасних технологій та вимог ринку.
2. Розгляд і аналіз існуючих систем та програм для обробки інформації про деталі шкіргалантереї з метою визначення прогалин та можливостей їх поліпшення.
3. Розробка та впровадження математичних моделей, алгоритмів та програмного забезпечення для автоматизованої підготовки інформації про шкіргалантерею.
4. Проведення експериментів, тестування та аналіз результатів для оцінки ефективності розробленого програмного продукту та порівняння його з існуючими системами.

Ця робота спрямована на поліпшення процесів підготовки інформації про шкіргалантерею та сприятиме подальшому розвитку галузі шкіргалантереї.

# РОЗДІЛ 1: Постановка проблеми автоматизованої підготовки інформації про деталі шкіргалантереї

## 1.1. Визначення та обґрунтування важливості автоматизованої підготовки інформації про деталі шкіргалантереї

Автоматизована підготовка інформації в шкіргалантереї – це процес використання технологій, програмного забезпечення та автоматизованих систем для оптимізації виробничих процесів у сфері виробництва швейних виробів. Цей процес включає в себе автоматизацію виробництва, контроль якості, управління запасами та планування виробництва.

Автоматизована підготовка інформації в легкій промисловості відіграє важливу роль у поліпшенні ефективності виробництва та конкурентоспроможності компаній. Ось декілька обґрунтувань, чому цей процес є важливим:

- Підвищення продуктивності: Автоматизація допомагає зменшити час виготовлення продукції, підвищити швидкість виробництва та знизити ризики людських помилок.
- Зниження витрат: Автоматизація дозволяє оптимізувати використання ресурсів та зменшити витрати на оплату праці.
- Підвищення якості: Системи автоматизації дозволяють здійснювати постійний контроль якості, що сприяє виробництву високоякісних продуктів.
- Управління запасами: Автоматизована система контролю запасів допомагає уникнути надмірного запасу та забезпечує наявність необхідних матеріалів.
- Гнучкість виробництва: Системи автоматизації дозволяють легше змінювати виробничі процеси та адаптуватися до ринкових змін.


Рисунок 1 – Автоматизований процес виробництва в шкіргалантереї

Вирішальне значення для виробництва красивих, добротних та дешевих виробів шкіргалантереї мають процеси моделювання та конструювання. Розробка конструкцій лекал, їх перевірка, уточнення в багатьох випадках виготовляється конструкторами, на яких в масовому виробництві покладаються складні і відповідальні задачі [1].

Впровадження автоматизації вимагає інвестицій та організаційних змін, але це сприяє зростанню конкурентоспроможності продукції на ринку. Загалом, автоматизація відіграє важливу роль у покращенні ефективності та сталості виробництва товарів, зменшенні негативного впливу на довкілля та забезпеченні якості продукції.

Додаткові елементи, які варто врахувати під час інвестування в розробку автоматизованих програм виробництва шкіргалантереї:

- Вплив на економію ресурсів;
- Покращення умов праці;
- Можливості для інновацій;
- Споживчий попит та конкурентоспроможність;
- Переваги для довкілля;
- Впровадження та інвестиції.

Адже основні перспективні напрями розвитку легкої промисловості України полягають в інтенсифікації зусиль вітчизняних виробників щодо завоювання стійких конкурентних позицій на внутрішньому ринку шляхом

поліпшення якості, підвищення наукоємності продукції, зниження її собівартості [2].

### 1.2. Мета та завдання дослідження

Мета та завдання дослідження роботи полягають у розв'язанні проблем, пов'язаних із збором, обробкою та аналізом інформації про вироби шкіргалантереї в промисловому виробництві. Ця галузь легкої промисловості має своєю особливістю велику кількість деталей та компонентів, що потребують систематичного моніторингу та управління. Основною метою дослідження є створення програмного забезпечення, яке дозволить автоматизувати процес підготовки інформації про деталі шкіргалантереї, зменшити людський вплив та покращити продуктивність виробництва.

Основні завдання дослідження включають:

1. Визначення та обґрунтування важливості автоматизованої підготовки інформації про деталі шкіргалантереї: Розкриття сутності проблеми та її впливу на ефективність виробництва шкіргалантереї.
2. Актуальність проблеми в галузі шкіргалантереї та автоматизації процесів: Визначення та обґрунтування того, чому автоматизація є актуальною в галузі шкіргалантереї та чому потрібно дослідження в цьому напрямку.
3. Огляд існуючих систем та програм для обробки інформації про деталі шкіргалантереї: Аналіз наявних рішень на ринку та визначення їх переваг та недоліків.
4. Розгляд математичних підходів та алгоритмів: Дослідження наукових методів та алгоритмів, які можуть бути використані для автоматизації підготовки інформації про деталі шкіргалантереї.
5. Розробка програмного забезпечення: Створення програмного продукту на основі обраних технологій та методів.

6. Експерименти та тестування: Проведення експериментів для валідації розробленого програмного забезпечення та порівняння результатів з існуючими системами.

7. Висновки та можливості подальшого розвитку: Узагальнення отриманих результатів, визначення можливостей подальшого вдосконалення та розвитку розробленого продукту.

Дослідження цієї роботи спрямоване на покращення ефективності управління процесами виробництва шкіргалантереї, що може призвести до зменшення витрат та підвищення якості виробів. Така автоматизація може бути важливою для підприємств, що діють в галузі легкої промисловості, де кількість та різноманітність деталей є значними факторами.

## 1.3. Актуальність проблеми в галузі шкіргалантереї та автоматизації процесів

Галузь шкіргалантереї, або легкої промисловості, є важливим сегментом економіки, де виробництво різноманітних швейних виробів відбувається на велику кількість деталей та компонентів. Завдяки своїй масовості та різноманітності, ця галузь стикається з численними викликами та проблемами, які стають актуальними в контексті автоматизації процесів виробництва.

Таблиця 1 – Класифікація предметів шкіргалантереї [3]

| Класифікаційна група | Перелік продукції |
|---|---|
| Сумки | • Сумки для жінок<br>• Сумки для чоловіків<br>• Портфелі<br>• Рюкзаки<br>• Сумки-месенджери<br>• Портфель-рюкзаки |
| Гаманці та гаманці для грошей | • Жіночі гаманці<br>• Чоловічі гаманці<br>• Карткові гаманці<br>• Монетниці<br>• Гаманці-клатчі |
| Ремені | • Чоловічі ремені<br>• Жіночі ремені |

| | |
|---|---|
| | • Ремені для джинсів<br>• Ремені для суконь |
| Портфелі та гаманці | • Чоловічі портфелі<br>• Жіночі кошельки<br>• Бізнес-портфелі<br>• Портмоне<br>• Гаманці для кредитних карток |
| Аксесуари | • Рукавички<br>• Шарфи<br>• Шапки<br>• Головні убори<br>• Зонтики |
| Чохли для гаджетів | • Чохли для смартфонів<br>• Чохли для ноутбуків<br>• Чохли для планшетів |
| Поясні сумки | • Спортивні поясні сумки<br>• Поясні сумки для подорожей<br>• Поясні гаманці |
| Взуття | • Чоловіче взуття<br>• Жіноче взуття<br>• Спортивне взуття<br>• Сапоги |
| Інші вироби шкіряної галантереї | • Перчатки<br>• Чехли для документів<br>• Гаманці для паспортів<br>• Аксесуари для сумок (ремінці, ланцюжки тощо) |

Однією з найбільш актуальних проблем є збільшення продуктивності та зниження витрат виробництва. Це особливо важливо в умовах конкурентного ринку та високого попиту на текстильні вироби. Автоматизація процесів, пов'язаних із підготовкою інформації про деталі шкіргалантереї, може допомогти знизити трудомісткість та підвищити продуктивність виробництва.

Другою важливою актуальною проблемою є точність та якість виробів. Шкіргалантерея часто включає в себе складні конструкції та декоративні елементи, тому точність обробки деталей має велике значення. Автоматизована система, яка може виявляти і усувати дефекти або недоліки на ранніх стадіях виробництва, може значно підвищити якість готових виробів та зменшити відхилення від заданих стандартів.

Третьою актуальною проблемою є впровадження нових технологій та стандартів у галузь шкіргалантереї. Швидкі зміни у модному світі та вимоги до виробів призводять до необхідності швидкої адаптації виробників до нових

трендів. Автоматизація дозволяє швидко впроваджувати зміни в процесах та виробництві, що є важливим фактором конкурентоспроможності в цій галузі.

Отже, актуальність проблеми автоматизації підготовки інформації про деталі шкіргалантереї полягає в необхідності підвищення продуктивності, якості виробів та адаптації до швидких змін у галузі.

Виробництво шкіряних, хутряних матеріалів та взуття пов’язане з шкідливим впливом на людину та екологічну обстановку оточуючого середовища, потребує першочергового впровадження засобів автоматизації. Їх застосування дозволяє суттєво вплинути на кінцеву якість продукції. А також змінити кардинально умови праці, забезпечивши чистоту цехів та відсутність специфічних для виробництва шкіри, хутра та взуття випаровувань хімікатів, тобто комфортних та безпечних умов праці для співробітників. Швейне та взуттєве виробництво включає етап крою деталей та безпосередньо виготовлення виробів. Етап крою є чинником, що формує подальшу якість продукції. Саме тому важливо забезпечення автоматизованого крою, що дозволяє уникнути помилок, пов’язаних з людським чинником та знизити кількість відходів під час крою. Виробництво взуття включає також операції, пов’язані з нагріванням з’єднаних деталей та їх витримкою для надійного кріплення. Автоматизація цих процесів дозволяє забезпечити надійність та довговічність взуття [4].

Розробка програмного забезпечення для автоматизації процесів має потенціал значно покращити виробництво та забезпечити стабільність підприємств в цій галузі.

### 1.4. Обґрунтування вибору даної теми

Збільшення обсягів виготовлення деталей та текстильних виробів вимагає від підприємств швидкої та ефективної обробки великої кількості інформації про ці деталі. Поточні методи та системи не завжди відповідають цим вимогам.

Обґрунтування даної теми полягає в розпізнанні актуальності автоматизації процесу підготовки інформації про деталі шкіргалантереї. Це дозволить підприємствам знизити трудомісткість, підвищити продуктивність та якість виробництва, а також швидше реагувати на зміни в ринкових умовах та модних трендах.

Крім того, існують прогалини в наявних системах та методах, які не завжди відповідають потребам галузі. Розробка нового підходу до автоматизації підготовки інформації про деталі шкіргалантереї може сприяти вирішенню цих проблем та покращити конкурентоспроможність підприємств в цій галузі.

### 1.5. Огляд існуючих систем та програм для обробки інформації про деталі шкіргалантереї

Велике поширення інформаційних технологій призвело до того, що зараз на більшості підприємств використовуються одночасно дві і більше системи автоматизованого проектування (САПР) з різним набором прикладних модулів CAD (англ. Computer-Aided Design) /CAM (англ. Computer-aided manufacturing) /CAE (англ. Computer-aided engineering) [5].

Огляд існуючих систем та програм для обробки інформації про деталі шкіргалантереї може допомогти зрозуміти, які рішення вже існують на ринку та які можливості вони пропонують. Нижче представлено загальний огляд таких систем та програм:

1. Gerber Technology AccuMark: Ця програма спеціалізується на розробці одягу та швейних виробів. Вона надає можливість створювати і аналізувати векторні шаблони деталей та розміщувати їх на матеріалі для подальшого виробництва. AccuMark також включає модулі для автоматичного розкрою матеріалу та оптимізації виробництва [6,7].
2. Optitex: Спеціалізується на віртуальному моделюванні та виробництві одягу. Optitex дозволяє створювати 3D-моделі виробів, використовуючи

векторні шаблони деталей та візуалізувати, як вони виглядають на реальних моделях. Це полегшує процес проектування та виробництва [8].

3. Tukatech TUKAcad: Програма призначена для проектування та розміщення шаблонів на матеріалі. Вона має інтуїтивний інтерфейс та набір інструментів для роботи із шаблонами деталей. TUKAcad також дозволяє створювати та зберігати цифрові архіви деталей для майбутнього використання [9].

4. Lectra Modaris: Спеціалізується на паттерн-конструюванні та градаціях розмірів. Вона дозволяє створювати векторні шаблони деталей та адаптувати їх до різних розмірів. Modaris також включає візуалізацію у 3D для аналізу пасформи та фітингу [10].

5. Browzwear VStitcher: Надає можливість створювати віртуальні прототипи одягу та додаткових виробів. BStitcher дозволяє виготовляти векторні шаблони та відразу перевіряти, як вони виглядають на 3D-моделях [11].

Використовуючи інструменти цифрового розкрою, дизайн носимих виробів може бути розроблений дуже швидко та ефективно. Можна вибирати різні матеріали, якщо вони доступні у бібліотеці програми, або імпортувати цифровий варіант конкретного матеріалу. Цей процес забезпечує численні варіанти дизайну для процесу прийняття рішень, ресурси не витрачаються марно, оскільки все розробляється в цифровому та віртуальному середовищі, і візуалізація цифрового носимого виробу дуже близька до фактичного фізичного виробу після його виготовлення [12].

Іноземний автор подав інформацію про різні інструменти для 3D моделювання наступним чином:

Таблиця 2 – Інструменти моделювання продукції [12]

| Назва | Опис |
|---|---|
| Accumark3D | Accumark3D - це повністю інтегрований 3D інструмент для системи Accumark2D CAD та YniquePLM. Подібно до V-Stitcher, Accumark3D використовує потужний відкритий симуляційний двигун Blender, який широко використовується в індустрії анімації, кіно, відеоігор та симуляції. Цей 3D інструмент для віртуального зразкування спрямований на допомогу компаніям у галузі одягу зменшити час та витрати на розробку та виготовлення зразків. Також можливо створювати 3D зображення, використовуючи параметричні, повністю налаштовувані матеріали з бібліотеки Substance Source, яка налічує понад 6000 матеріалів. |
| CLO3D | CLO3D - це комерційний рішення, яке дозволяє створювати віртуальний процес примірки, вводячи 2D малюнки та віртуально шиючи їх на 3D цифрову модель людини (аватара). Користувачі можуть візуалізувати примірку гардеробу в 3D на етапі малювання. Ця платформа також включає багатий каталог більше ніж 900 "цифрових клонів" фізичних тканин для мінімізації відходів. |
| Lectra Modaris 3D | Modaris classic та Modaris 3D використовуються на всіх етапах розробки малюнків, починаючи від початкової цифровизації до 3D віртуального прототипування. Програмне забезпечення Modaris Pattern Cutting є лідером у рішеннях від Lectra та інтегрується на одній платформі з Modaris 3D Virtual Try-on для 2D малюнків одягу. Подібно до інших 3D інструментів, згаданих у цьому розділі, цей також включає бібліотеки даних, специфічні для галузі, з більш ніж 300 зразками матеріалів. |
| Optitex3D | Pattern Design Software (PDS) 3D - це назва 3D віртуального генератора зразків від Optitex, повністю інтегрованого з PDS 2D цифровим рішенням зразків. Користувач може створювати нові малюнки, редагувати існуючі з бази даних або імпортувати малюнок з іншої системи у форматах .dxf, .asthma та .aama. Також система надає цифрову бібліотеку тканин, але користувач може вимірювати та моделювати нові тканини в 3D на основі їх фізичних і візуальних властивостей. |
| Style3D | Style3D - це 3D інструмент для керування тканинами, створений Linctex. Управління тканинами здійснюється через рішення Style 3D, де існуючі матеріали скануються для створення фотореалістичних цифрових зразків, які можуть бути використані для тих же цілей, що і цифрово розроблені тканини. |
| V-Stitcher (Browzwear) | V-Stitcher від Browzwear - цифровий інструмент для розробки моди в 3D, спрямований на модельєрів малюнків, крою і технічних дизайнерів. Рішення Lotta підходить для дизайнерів, V-Stitcher - для модельєрів і виробників, а Stylezone - це хмарна платформа для демонстрації 3D дизайнів в інтернеті та на мобільних пристроях. Browzwear's Fabric Analyzer (FAB) - це частина розширюються цифрової екосистеми, яка надає користувачам здатність визначати всі фізичні властивості будь-якого матеріалу, від його товщини до розтягування та згину. |
| Tuka3D (Tukatech) | Tuka3D - це система віртуального прототипування від Tukatech. Вона надає індивідуальні віртуальні моделі підгону та створює життєподібні віртуальні зразки одягу. Новизною у версії 2022 року є можливість використовувати відкриту систему, яка дозволяє дизайнерам, брендам, роздрібним продавцям та їх фабрикам працювати ефективно в рамках віртуального процесу. Це означає, що користувачі інших 3D систем можуть розпочати свій робочий процес з 266 реальних реплік аватарів з великої бібліотеки Tukatech, що налічує понад 700 моделей. |

Але особливу увагу варто приділити і ціні програмного продукту, адже це є частиною бюджетування та собівартості.

| 1. GERBER TECHNOLOGY ACCUMARK | |
|---|---|
| **Доступність:**<br>Вартість та ліцензійні умови можуть варіюватися. Зазвичай, ця програма пропонує користувачам опції для одного або декількох користувачів. | **Вартість:**<br>Від декількох тисяч доларів за ліцензію. |
| 2. OPTITEX | |
| **Доступність:**<br>Зазвичай, ця програма доступна для користувачів індивідуально, а також для команд та підприємств. | **Вартість:**<br>Коливатися від декількох тисяч доларів за одиничну ліцензію до більших сум для комерційних користувачів. |
| 3. TUKATECH TUKACAD | |
| **Доступність:**<br>Зазвичай доступна для індивідуальних користувачів та | **Вартість:**<br>Вартість ліцензії може бути від декількох сотень до декількох тисяч доларів. |
| 4. BROWZWEAR VSTITCHER | |
| **Доступність:**<br>Доступна для індивідуальних користувачів та команд. | **Вартість:**<br>Може бути від декількох тисяч до десятків тисяч доларів. |

Рисунок 2 – Огляд програм для обробки інформації про деталі шкіргалантереї та їхні ліцензійні умови [6-11]

Огляд існуючих систем показує, що є багато програм, які можуть бути корисні в галузі шкіргалантереї. Вибір конкретної системи залежить від специфічних потреб підприємства та процесів, які воно прагне автоматизувати.

### 1.6. Методи дослідження та основні етапи розробки

Методи дослідження та основні етапи цієї роботи включали наступне:

Методи дослідження які були використані:

- Аналіз літератури: Проведення огляду наукових статей, книг та інших джерел, щоб дізнатися про попередні дослідження та відомості в обраній галузі.

- Емпіричні дослідження: Збір даних через спостереження, опитування або експерименти, щоб дослідити конкретні аспекти автоматизації в легкій промисловості.

Методи дослідження які необхідно використовувати в подальших дослідженнях:

- Кейс-стаді: Детальний аналіз конкретних підприємств або виробничих ліній, які вже використовують автоматизовані системи.
- Опитування та інтерв'ю: Спілкування з фахівцями та працівниками, які мають досвід впровадження автоматизації в легкій промисловості.

Основні етапи цієї роботи представлено в наступному рисунку.

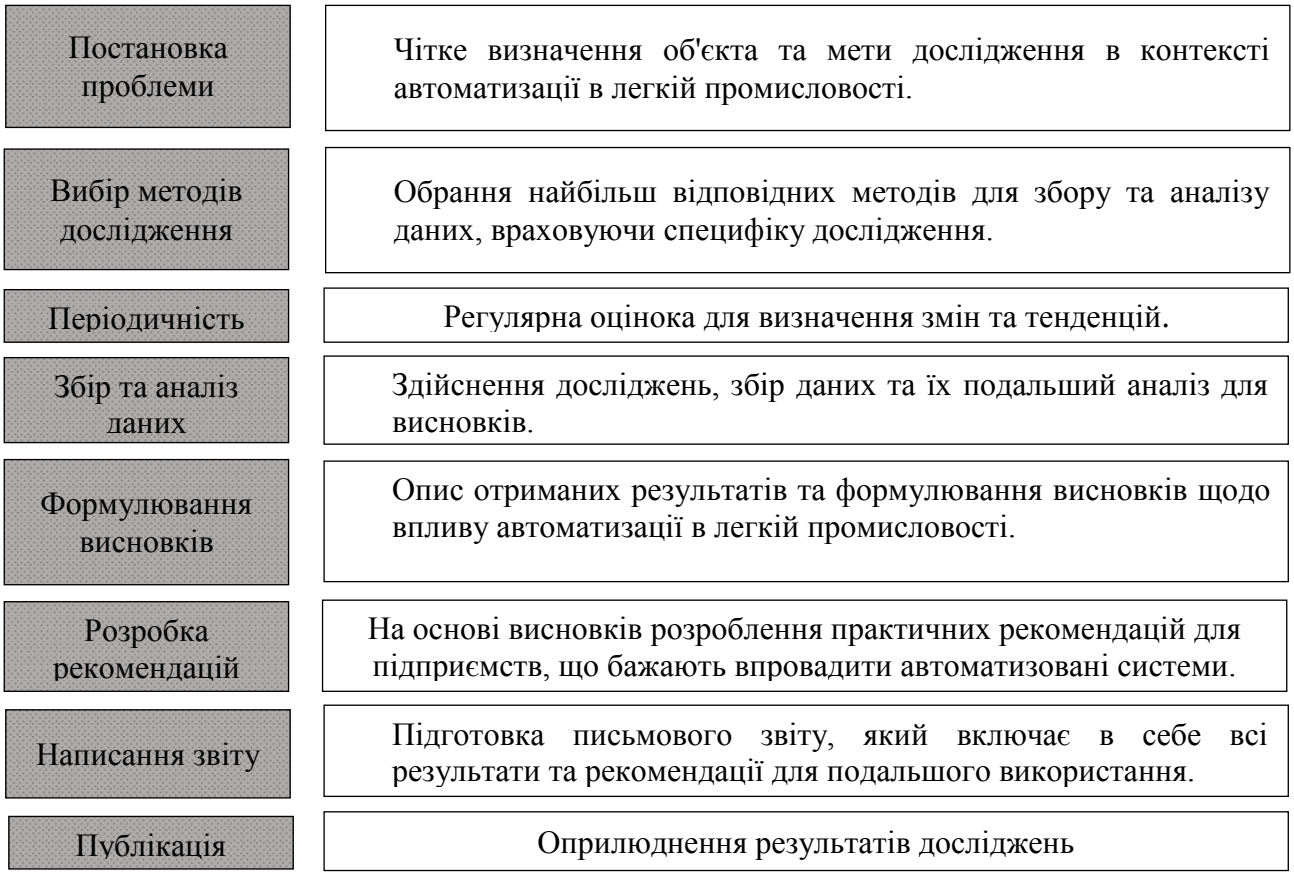



Рисунок 3 – Головні етапи

Ці методи та етапи допомагають систематично дослідити важливі аспекти автоматизації в виробництві шкіргалантереї та надати практичні рекомендації для підприємств сектору легкої промисловості.

## Висновки до розділу 1

Дослідження впровадження автоматизації в легкій промисловості є важливим завданням, оскільки цей сектор великого значення для економіки та суспільства загалом.

Можна зробити висновок, що автоматизація має потенціал покращити ефективність виробництва, зменшити витрати ресурсів, покращити умови праці працівників, сприяти інноваціям та збільшити конкурентоспроможність підприємств у легкій промисловості

Важливо підкреслити, що автоматизація вимагає інвестицій та ретельного планування, а також постійного моніторингу та оновлення, щоб враховувати зміни в галузі та ринковому середовищі. Постійний аналіз і оновлення є ключовими елементами успішної реалізації автоматизованих систем в легкій промисловості.

Дослідження проводиться з метою з'ясування, як в промисловості виготовлення шкіргалантереї можна впровадити автоматизаційні системи для підвищення продуктивності. Аналіз компаній які створюють схоже програмне забезпечення може бути корисним для ідентифікації кращих практик, а також виявлення можливих труднощів і викликів у процесі впровадження автоматизації.

# РОЗДІЛ 2: Теоретичний огляд

## 2.1. Огляд літератури щодо процесів шкіргалантереї та існуючих методів обробки інформації

Є стандарти, які допомагають виробникам та споживачам гарантувати якість та безпеку продукції в галузі шкіргалантереї. Підприємства, які дотримуються цих стандартів, можуть позначати свої вироби як відповідні вимогам безпеки та якості. Ось деякі з найбільш важливих стандартів, пов'язаних із шкіргалантереєю:

1. **ISO 14001:** Стандарт, що встановлює вимоги до системи управління навколишнього середовища. Важливий для виробників, які прагнуть зменшити негативний вплив свого виробництва на навколишнє середовище [13].
2. **ISO 9001:** Стандарт якості, який регулює системи управління якістю. Важливий для забезпечення якості виробів шкіргалантереї та задоволення потреб клієнтів [14].
3. **REACH:** Регуляція Європейського Союзу, яка стосується реєстрації, оцінки, авторизації та обмеження хімічних речовин. Важлива для виробників шкіргалантереї, оскільки виробництво може включати використання хімічних речовин [15].
4. **OEKO-TEX Standard 100:** Сертифікаційний стандарт, який визначає вимоги до текстильних виробів, включаючи текстиль у шкіргалантереї, щодо вмісту шкідливих речовин та інших критеріїв безпеки [16].
5. **ASTM International:** Організація, яка розробляє та встановлює стандарти для широкого спектру продуктів, включаючи матеріали та методи випробувань, що можуть бути застосовані в галузі шкіргалантереї [17].

Існують іноземні джерела, пов'язані з автоматизацією виробництва у галузі шкіргалантереї та текстильної промисловості.

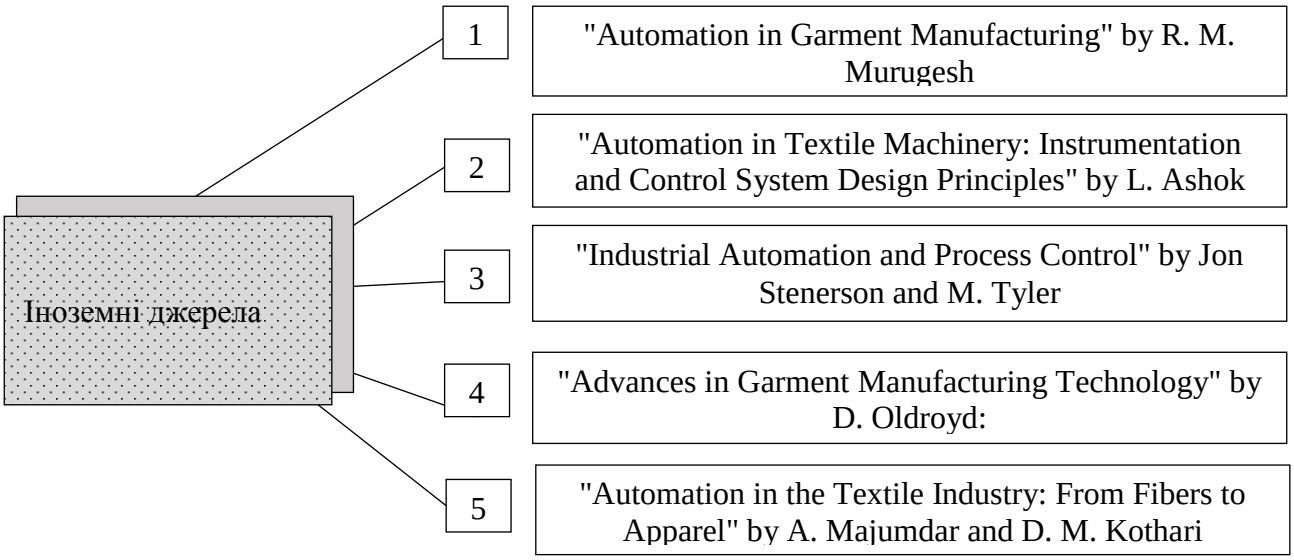



Рисунок 4 – Іноземні джерела, пов'язані з автоматизацією виробництва у галузі шкіргалантереї та текстильної промисловості

В Україні також існують стандарти і нормативні документи, пов'язані із шкіргалантереєю та текстильним виробництвом. Наприклад, Державний комітет України з питань технічного регулювання та споживчої політики (ДКТРСП) розробляє та впроваджує стандарти якості та безпеки продукції в Україні. Державні стандарти України регулюють багато аспектів текстильного та швейного виробництва, включаючи вимоги до якості та безпеки продукції.

Також є санітарні норми і правила, які стосуються безпеки та якості текстильної продукції, зокрема виробництва одягу та текстильних матеріалів.

## 2.2. Розгляд математичних підходів та алгоритмів, які можуть бути застосовані до автоматизації підготовки інформації

Наступні математичні підходи та алгоритми можуть бути інтегровані в системи автоматизації виробництва для полегшення процесу підготовки інформації, контролю якості, планування та оптимізації виробництва в галузі шкіргалантереї:

1. Математична обробка зображень: Для якісного контролю якості та аналізу дефектів виробів шкіргалантереї можна використовувати методи обробки зображень. Алгоритми для виявлення дефектів, вимірювання розмірів

та оцінки колірної якості можуть використовуватися для автоматизованої ідентифікації та класифікації виробів.

2. Математичні моделі якості: Розроблення математичних моделей для оцінки якості шкіргалантереї може допомогти автоматизувати процес контролю. Моделі можуть базуватися на параметрах, таких як міцність матеріалу, геометрія, інші характеристики, і допомагати приймати рішення щодо придатності продукції.

3. Оптимізація виробництва: Математичні алгоритми для оптимізації процесів виробництва можуть допомогти знизити витрати та підвищити продуктивність. Це може включати розподіл ресурсів, планування виробництва, оптимізацію ланцюга постачання тощо.

4. Автоматизована ідентифікація та відстеження: Математичні алгоритми розпізнавання об'єктів та машинне навчання можуть використовуватися для ідентифікації і відстеження виробів, що спрощує процес логістики та контролю якості.

5. Прогнозування виробництва: Методи прогнозування, такі як часові ряди та аналіз даних, можуть бути застосовані для передбачення попиту на вироби шкіргалантереї та оптимізації виробництва відповідно до попиту.

6. Автоматизована обробка тексту: Якщо підготовка інформації включає в себе обробку текстової інформації, можна використовувати алгоритми обробки природної мови (NLP) для автоматичного аналізу і категоризації текстових даних.

7. Математичне моделювання процесів виробництва: Системи моделювання можуть бути використані для оптимізації та вдосконалення процесів виробництва в галузі шкіргалантереї.

Програмування в цьому випадку відіграє важливу роль у впровадженні математичних підходів і алгоритмів в автоматизацію підготовки інформації в галузі шкіргалантереї. Розробники програмного забезпечення використовують математичні методи та алгоритми для створення систем, які здатні опрацьовувати великі обсяги даних та автоматизувати рутинні процеси.

Програмування включає в себе реалізацію математичних алгоритмів в програмному коді, що дозволяє їх використання у реальному часі. Деякі мови програмування, такі як Python, R, чи Java, надають потужні інструменти для математичного моделювання та аналізу даних.

У контексті автоматизації підготовки інформації для галузі шкіргалантереї, програмування може бути використане для реалізації наступних завдань:


Рисунок 5 – Використання програмного забезпечення в автоматизації виготовлення шкіргалантереї

Програмування створює можливість інтегрувати математичні підходи і алгоритми в програмне забезпечення, що спрощує і поліпшує процеси автоматизації підготовки інформації в галузі шкіргалантереї.

## 2.3. Вибір та обґрунтування використовуваних технологій та інструментів програмування

Для розробки програмного забезпечення в галузі автоматизації підготовки інформації для шкіргалантереї, можна використовувати різноманітні інструменти та технології, залежно від конкретних завдань та потреб проєкту. Ось деякі з них:

1. Мови програмування:
    - Python: Популярна мова для розробки програмного забезпечення та використання в аналізі даних.
    - Java: Для розробки великих та високопродуктивних додатків.
    - R: Особливо корисний для статистичного аналізу та обробки даних.
2. Бібліотеки та фреймворки:

- TensorFlow та PyTorch: Для розробки моделей машинного навчання.
- OpenCV: Для обробки зображень та розпізнавання об'єктів.
- Flask та Django: Для розробки веб-додатків.

3. Інструменти обробки зображень:
    - Adobe Photoshop: Для ручної обробки та ретуші зображень.
    - GIMP: Безкоштовний редактор зображень.
4. Системи управління базами даних (СУБД):
    - MySQL, PostgreSQL, SQLite: Для зберігання та керування даними.
    - MongoDB: Для зберігання та обробки неструктурованих даних.
5. Інструменти для роботи з текстовою інформацією:
    - Natural Language Toolkit (NLTK): Для обробки тексту та аналізу природної мови.
    - spaCy: Для розпізнавання та аналізу текстових даних.
6. Інструменти для автоматизації процесів:
    - Robotic Process Automation (RPA) інструменти, такі як UiPath та Automation Anywhere: Для автоматизації рутинних завдань та процесів.
7. Системи моделювання та оптимізації:
    - MATLAB: Для математичного моделювання та оптимізації процесів.
    - AnyLogic: Для моделювання систем і оптимізації виробничих процесів.
8. Інструменти для автоматизації виробництва:
    - Manufacturing Execution Systems (MES): Для керування та контролю виробництвом.

Це лише декілька прикладів інструментів, які можна використовувати в програмуванні для автоматизації підготовки інформації в галузі шкіргалантереї. Вибір конкретних інструментів залежить від потреб проєкту, характеру даних та завдань, які необхідно вирішити.

Завдання виробництва шкіргалантереї вимагає комплексного підходу та ефективного управління різними аспектами бізнесу. Неможливо узагальнити всі підприємства для аналізу. Тому для майбутнього варто вибрати конкретну продукцію яка розробляється підприємством та розглянути потребу в програмному забезпеченні для створення цього продукту.

Для прикладу можна взяти підприємство з виробництва шкіряних сумок.

На рис. зображено потреби виробника цього виду продукту:

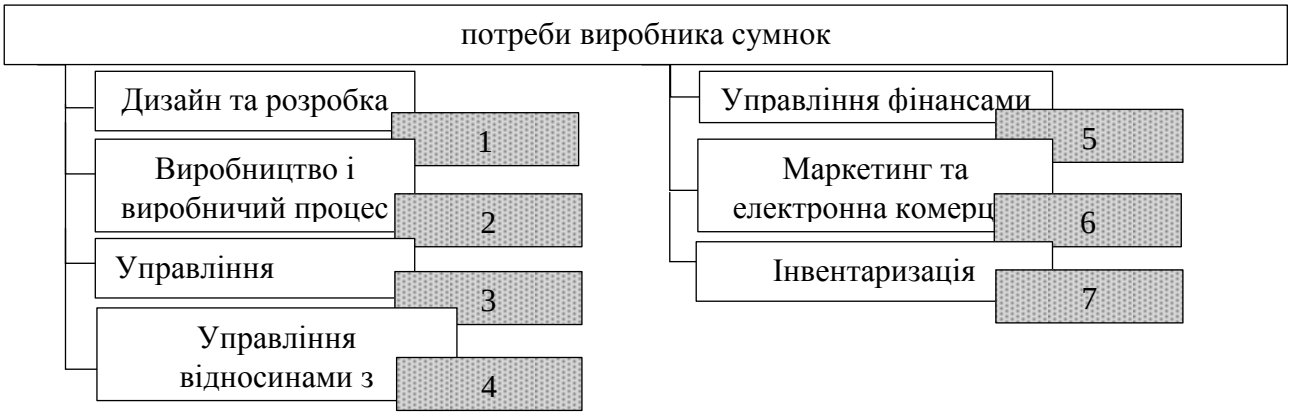



Рисунок 6 – Основні потреби виробника шкірогалантерейної галузі

Абсолютно можливе створення програмного забезпечення (ПЗ), яке об'єднає різні функціональність для виробництва шкіргалантереї у єдину систему. Такі комплексні програми, як правило, називаються Enterprise Resource Planning (ERP) системами або Manufacturing Resource Planning (MRP) системами. Вони поєднують різні модулі і функціональність для ефективного управління всіма аспектами виробництва та бізнесу.

Створення комплексних систем управління ресурсами підприємства (ERP) або систем управління виробництвом (MRP) є складним завданням і може вимагати значних зусиль та ресурсів. Необхідна команда розробників, які розуміють процеси виробництва та бізнес-процеси окремої компанії.

Але цілком можливо звести функціональність до мінімуму таким чином спростивши процес, залишивши головні структурні елементи.

Створення програми розкрою і управління для виробників шкіргалантерейних виробів, яку розробляє одна особа, може бути складним завданням, але основний функціонал може включати такі можливості:

Таблиця 3 – Можливості розроблювальної програми

| **Функціонал** | **Опис** | **Чи включається в майбутню програму** |
|---|---|---|
| *Дизайн і розкрій візерунків* | Створення та редагування візерунків для продукції. | - |
| *Управління матеріалами і запасами* | Ведення обліку та керування запасами сировини та компонентів. | + |
| *Списання матеріалів* | Запис списання матеріалів після розкрою візерунків. | + |
| *Управління замовленнями та виробництвом* | Приймання та відстеження замовлень на виготовлення продукції. | - |
| *Фінансовий облік* | Облік витрат на матеріали, працю та інші витрати. | - |
| *Генерація звітів* | Можливість генерації звітів про виробництво, розкрій матеріалів та фінанси. | + |
| *Інтеграція з обліковими програмами* | Імпорт та експорт даних для обліку та фінансового обліку. | + |
| *Керування клієнтами* | Ведення обліку клієнтів, їх замовлень та історії покупок. | - |
| *Безпека даних* | Захист і резервне копіювання даних для запобігання втратам. | - |
| *Інтуїтивний інтерфейс* | Зручний та легкий у використанні інтерфейс для користувача. | + |
| *Підтримка кількох мов і валют* | Можливість робити бізнес на міжнародному рівні. | - |
| *Мобільний доступ* | Можливість керувати виробництвом та перевіряти статус замовлень з мобільного пристрою. | - |
| *Оновлення і підтримка* | Постійні оновлення та можливість отримувати підтримку від розробника. | - |

Таким чином можна максимально спростити функціонал для майбутнього програмного забезпечення.

Для проєкту з розробки програмного забезпечення для розкреслення деталей шкіргалантереї, який має обмежений бюджет і вимагає легкості використання, рекомендується такі конкретні інструменти: HTML та JavaScript. Ці інструменти безкоштовні, легко доступні для вивчення та використання, та надають засоби для обробки та візуалізації даних.

## 2.4. Порівняння попередніх рішень та визначення прогалин

При розробці програмного забезпечення (ПЗ) для розкрою матеріалу, такого як шкіра для галантереї, можуть виникнути різні прогалини або проблеми, які варто враховувати:

1. Нестача точності в розкрої: Однією з основних проблем є необхідність точного розкрою матеріалу, щоб мінімізувати відходи та витрати матеріалу.
2. Складність моделей: Якщо ваші вироби мають складні форми або деталі, розкрій матеріалу може бути складним завданням, особливо якщо ви шукаєте оптимальний варіант розміщення деталей.
3. Врахування втрат під час розкрою: Під час розкрою шкіри може виникати втрати матеріалу через обрізку, шви та інші технологічні процеси.
4. Оптимізація виробництва: Програмне забезпечення для розкрою матеріалу повинно оптимізувати процеси виробництва, враховуючи обмеження обладнання, робочої сили та інші фактори.
5. Масштабованість: ПЗ повинно бути готове для масштабування для роботи з різними розмірами виробів та кількостями.
6. Інтеграція з іншими системами: Якщо ви ведете виробництво галантерей, важливо, щоб ПЗ було інтегроване з іншими системами, такими як управління запасами та облік.
7. Приватність та безпека: Якщо ви зберігаєте конфіденційну інформацію про свої вироби та клієнтів, важливо забезпечити безпеку та захист даних.
8. Підтримка та навчання: Користувачам може знадобитися підтримка та навчання по роботі з ПЗ.
9. Оновлення та підтримка: ПЗ повинно регулярно оновлюватися для виправлення помилок та додавання нових функцій.

10. Вартість: Вартість розробки та впровадження ПЗ може бути високою, і важливо визначити, чи вона виправдовується витратами.

Загалом, розробка ПЗ для розкрою матеріалу шкіргалантереї вимагає уважного розгляду всіх аспектів виробництва, включаючи точність, оптимізацію виробництва та ефективність витрат матеріалу.

## Висновки до розділу 2

У процесі огляду літератури було розглянуто значення автоматизованої підготовки інформації для галузі шкіргалантереї. Літературні джерела надали цінний інсайт щодо основних завдань та важливості використання програмного забезпечення для розкреслення деталей та автоматизації процесів виробництва.

Під час розгляду математичних підходів та алгоритмів було визначено, що системи моделювання можуть бути використані для оптимізації та вдосконалення виробничих процесів в галузі шкіргалантереї. Застосування методів машинного навчання та обробки зображень дозволяє автоматизувати процес розкреслення деталей та підвищити продуктивність.

У виборі інструментів для розробки програмного забезпечення було запропоновано конкретні інструменти, які відповідають вимогам проєкту та обмеженням бюджету.

Нарешті, було визначено виклики та ризики, пов'язані з розробкою програмного забезпечення для деталей шкіргалантереї. До них належать якість вихідних даних, великий обсяг даних, вартість обладнання, забезпечення безпеки, легальні питання та інші. Для успішної розробки проєкту необхідно враховувати ці виклики та вживати заходи для їх подолання.

Узагальнюючи, програмне забезпечення для розкреслення деталей шкіргалантереї має великий потенціал для оптимізації та автоматизації виробничих процесів. З врахуванням правильного вибору інструментів, методів та управління ризиками, цей проєкт може стати важливим кроком у вдосконаленні галузі шкіргалантереї та підвищенні її ефективності.

# РОЗДІЛ 3: Розробка та дослідження

## 3.1. Опис математичних моделей та алгоритмів для автоматизованої підготовки інформації про деталі шкіргалантереї

Необхідно розробити програму для розкрою деталей шкірогалантереї. Це має бути спрощена програма для працівників легкої промисловості, яке спеціалізується на виробництві сумок. Ця програма має робити схеми, розраховувати собівартість, використання матеріалу і генерувати звіти для керівництва. Також вона має бути легка в використанні.

Звужені інструменти для розробки програми для розкрою деталей шкірогалантереї включають наступне: HTML та JavaScript.

Ці інструменти дозволять розробити програму для розкрою деталей шкірогалантереї з деяким обмеженням функціоналу. Це дозволить зобразити механіку та основні алгоритми побудови програми.

HTML (Hypertext Markup Language) та JavaScript є ключовими інструментами у веб-розробці, які працюють разом, щоб створити динамічні та взаємодіючі веб-сайти. HTML визначає структуру та маркування веб-сторінок, тоді як JavaScript забезпечує можливості взаємодії та динаміки.

HTML є основною мовою розмітки для веб-сторінок. Вона використовує теги для визначення елементів на сторінці, таких як заголовки, абзаци, таблиці та зображення. HTML забезпечує основний каркас сторінки, але він стає більш динамічним завдяки JavaScript.

JavaScript є скриптовою мовою програмування, яка використовується для надання веб-сторінкам інтерактивності та динаміки. Вона може контролювати та змінювати HTML та CSS на сторінці. JavaScript може бути використаний для обробки подій, які сприяють взаємодії користувача з веб-сторінкою.

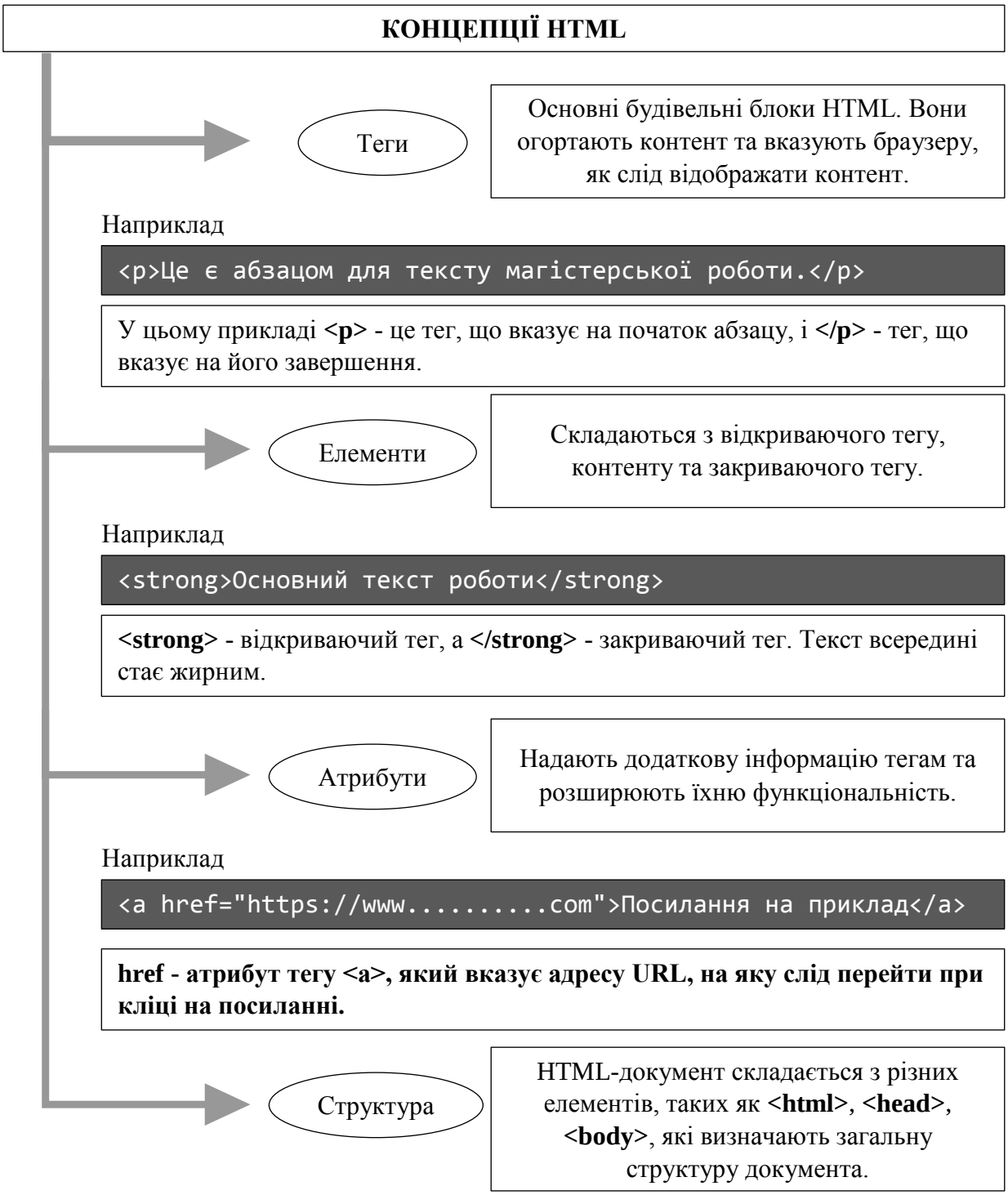



Рисунок 7 – Основні концепції HTML-документа

Java є мовою програмування, яка визначається своєю об'єктно-орієнтованою природою та можливістю переносити код на різні платформи. варто розглянути основні концепції, які лежать в основі програмування на Java:

Вони висвітлені на рис. 8.

Java - це об'єктно-орієнтована мова програмування, яка працює на різних платформах завдяки концепції "Write Once, Run Anywhere" (пиши один раз,

**Об'єктно-орієнтоване програмування (ООП):**

- **Класи та об'єкти:** Класи є шаблонами для створення об'єктів, які є екземплярами цих класів.
- **Інкапсуляція:** Засоби об'єднання даних та методів, що працюють з цими даними, в єдиному об'єкті.
- **Наслідування:** Можливість створювати новий клас на основі вже існуючого, успадковуючи його властивості та методи.
- **Поліморфізм:** Здатність об'єктів одного класу використовувати методи іншого класу.

**Синтаксис Java:**

- **Змінні та типи даних:** Визначення та використання змінних різних типів даних.
- **Оператори:** Використання математичних, порівняння та логічних операторів.
- **Керуючі конструкції:** Використання умовних операторів (if-else), циклів (for, while) та інших конструкцій управління потоком програми.

**Пакети та Модулі:**

- **Пакети:** Групування класів в пакети для організації коду та уникнення конфліктів імен.
- **Модулі (з Java 9):** Якщо розглядати нові версії Java, вони включають підтримку модульності, що дозволяє структурувати програму на більш великому рівні ніж пакети.

**Винятки та Обробка Помилок:**

- **Винятки (Exceptions):** Механізм для обробки помилок під час виконання програми.
- **Обробка винятків:** Використання блоків try, catch, finally для обробки винятків.

**Потоки (Streams) та Многозадачність:**

- **Потоки:** Взаємодія програми з введенням/виведенням та обробкою потоків даних.
- **Многозадачність:** Використання потоків для кількох завдань.

Рисунок 8 – Основні концепції Java

Взаємодія між Java та HTML відкриває безліч можливостей для розробки веб-додатків.

Java Applet - це програма, написана на мові програмування Java, яка вбудовується безпосередньо в HTML-документ та виконується на клієнтському браузері. Використання Java Applets дозволяє створювати графічно багаті та інтерактивні компоненти на веб-сторінці.

JavaServer Pages (JSP) - це технологія, яка дозволяє вставляти Java-код безпосередньо в HTML-сторінку. JSP використовується для створення динамічного вмісту на стороні сервера. У цьому випадку, Java-код використовується для генерації HTML-коду, який відправляється клієнту.

Java може бути використана для обробки даних, введених користувачем через HTML-форми. Наприклад, зчитування та обробка введених даних, валідація та відправка результатів на сервер для подальшої обробки. Також Java може генерувати HTML-код динамічно. Це може бути корисно, наприклад, при створенні списків, таблиць чи інших вмістових елементів, які залежать від даних, отриманих з бази даних або інших джерел.

Необхідно враховувати аспекти безпеки при взаємодії між Java та HTML, зокрема при використанні Java Applets. Забезпечення безпеки передбачає контроль за даними, які надходять від клієнтів, та захист від потенційних атак.

Безпека взаємодії між Java та HTML є критичним аспектом при розробці веб-додатків. Некоректно реалізована взаємодія може призвести до серйозних проблем, таких як атаки злому безпеки, втрата даних, або недостатній контроль доступу.

### 3.2. Розробка програмного забезпечення на базі обраної технології

Розробка цієї програми в цілому – це складне завдання, і воно може вимагати значної кількості часу та зусиль. Тому автор пропонує розділити роботу на декілька концептуально різних етапів:

*1. Схеми розкрою.*

```
<html>
<head>
    <title>Розрахунок розташування деталей</title>
    <style>
        #canvas {
            border: 1px solid blue;
            position: relative;
        }

        .detail {
            position: absolute;
            border: 1px solid red;
        }
    </style>
</head>
<body>
    <h1>Розрахунок розташування деталей на полотні</h1>

    <form id="inputForm">
        <h2>Полотно</h2>
        <label for="canvasWidth">Ширина полотна (см):</label>
        <input type="number" id="canvasWidth" required>

        <label for="canvasHeight">Висота полотна (см):</label>
        <input type="number" id="canvasHeight" required>

        <h2>Деталь №1</h2>
        <label for="detail1Width">Ширина деталі №1 (см):</label>
        <input type="number" id="detail1Width" required>

        <label for="detail1Height">Висота деталі №1 (см):</label>
        <input type="number" id="detail1Height" required>

        <label for="detail1Count">Кількість деталей №1:</label>
        <input type="number" id="detail1Count" required>

        <h2>Деталь №2</h2>
        <label for="detail2Width">Ширина деталі №2 (см):</label>
        <input type="number" id="detail2Width" required>

        <label for "detail2Height">Висота деталі №2 (см):</label>
        <input type="number" id="detail2Height" required>

        <label for="detail2Count">Кількість деталей №2:</label>
        <input type="number" id="detail2Count" required>

        <button type="button" id="calculateButton">Розрахувати</button>
    </form>

    <div id="canvas"></div>

    <script>
        document.addEventListener("DOMContentLoaded", function() {
            const inputForm = document.getElementById("inputForm");
            const calculateButton = document.getElementById("calculateButton");
            calculateButton.addEventListener("click", calculateLayout);
        });

        function calculateLayout() {
            const canvasWidth = parseFloat(document.getElementById("canvasWidth").value);
            const canvasHeight = parseFloat(document.getElementById("canvasHeight").value);

            const detail1Width = parseFloat(document.getElementById("detail1Width").value);
            const detail1Height = parseFloat(document.getElementById("detail1Height").value);
            const detail1Count = parseInt(document.getElementById("detail1Count").value);

            const detail2Width = parseFloat(document.getElementById("detail2Width").value);
            const detail2Height = parseFloat(document.getElementById("detail2Height").value);
            const detail2Count = parseInt(document.getElementById("detail2Count").value);
```

```
            const canvas = document.getElementById("canvas");
            canvas.style.width = canvasWidth + "cm";
            canvas.style.height = canvasHeight + "cm";
            canvas.innerHTML = "";

            const details = [];
            for (let i = 0; i < detail1Count; i++) {
                details.push({ width: detail1Width, height: detail1Height });
            }
            for (let i = 0; i < detail2Count; i++) {
                details.push({ width: detail2Width, height: detail2Height });
            }

            const packer = new RectanglePacker(canvasWidth, canvasHeight);
            const packedRectangles = packer.pack(details);

            for (const rect of packedRectangles) {
                const detail = document.createElement("div");
                detail.className = "detail";
                detail.style.width = rect.width + "cm";
                detail.style.height = rect.height + "cm";
                detail.style.left = rect.x + "cm";
                detail.style.top = rect.y + "cm";
                canvas.appendChild(detail);
            }
        }

        class RectanglePacker {
            constructor(width, height) {
                this.width = width;
                this.height = height;
                this.rectangles = [{ x: 0, y: 0, width, height }];
            }

            pack(rectangles) {
                const result = [];

                for (const rect of rectangles) {
                    for (const bin of this.rectangles) {
                        if (bin.width >= rect.width && bin.height >= rect.height) {
                            result.push({  x:  bin.x,  y:  bin.y,  width:  rect.width,  height:
rect.height });

                            if (bin.width > rect.width) {
                                this.rectangles.push({ x: bin.x + rect.width, y: bin.y, width:
bin.width - rect.width, height: rect.height });
                            }
                            if (bin.height > rect.height) {
                                this.rectangles.push({ x: bin.x, y: bin.y + rect.height, width:
bin.width, height: bin.height - rect.height });
                            }

                            this.rectangles.splice(this.rectangles.indexOf(bin), 1);
                            break;
                        }
                    }
                }

                return result;
            }
        }
    </script>
</body>
</html>
```

Ця програма вже має функціонал для розрахунку розташування деталей на полотні, але її необхідно поліпшити наступним чином:

1. Обробка помилок: Додавання обробки помилок допоможе уникнути виникнення непередбачуваних ситуацій. Наприклад, перевірка, чи всі необхідні поля заповнені коректно.

2. Анімація та візуалізація: Додати анімацію для покращення візуалізації процесу розташування деталей на полотні. Наприклад, анімовані переходи при додаванні деталей на полотно.

3. Збереження та завантаження стану: Додати можливість зберігати і завантажувати стан програми, щоб користувачі могли зберігати свої розрахунки та продовжувати їх пізніше.

4. Оптимізація алгоритму: Використовувати більш оптимізованих алгоритмів для розташування деталей на полотні, які мають більшу ефективність у випадку великої кількості деталей.

5. Розширення функціоналу: Додати можливість видалення деталей, зміни їх розмірів і позицій після розташування, а також інші функції, які зроблять програму більш потужною та корисною.

6. Адаптивність і мобільна підтримка: Аптувати програму до різних розмірів екранів та мобільних пристроїв.

Можна розглянути структуру та функціонал коду:

HTML та CSS частина:

| | |
|---|---|
| Структура форми | Форма містить поля для введення параметрів полотна та розмірів двох різних деталей. |
| Canvas та стилі | Елемент canvas використовується як візуалізація для розташування деталей. Стилі для canvas та .detail задають вигляд полотна і деталей відповідно. |

JavaScript частина

| | |
|---|---|
| Обробник подій при завантаженні | При завантаженні сторінки додається обробник подій, який прив'язується до кнопки "Розрахувати". |
| Функція | Ця функція отримує введені користувачем значення, |

| | |
|---|---|
| calculateLayout | створює об'єкти деталей, викликає функцію pack для розрахунку розташування деталей на полотні, та відображає їх на екрані. |
| Клас RectanglePacker | Цей клас відповідає за розташування прямокутників на полотні. Він використовує простий алгоритм упаковки прямокутників. |

Як код працює

1. Користувач вводить параметри полотна та двох різних деталей.
2. При кліці на кнопку "Розрахувати", викликається функція calculateLayout.
3. Функція створює об'єкти деталей та викликає алгоритм упаковки для розташування їх на полотні.
4. Розташовані деталі відображаються на canvas.

Зауваження

1. Оптимізація та обробка помилок: Код може бути оптимізований, і відсутні обробка помилок при введенні користувача. Додавання додаткової валідації та повідомлень про помилки покращило б досвід користувача.
2. Безпека вводу: Застосування безпечних методів для отримання та обробки введення користувача, щоб уникнути можливих атак.
3. Взаємодія з користувачем: Відсутність інтерфейсу для видалення або редагування деталей після розрахунку.

Таблична форма надає зручний огляд сильних та слабких сторін програми, а також можливостей та загроз, що можуть впливати на її розвиток та успіх. SWOT-аналіз допомагає визначити напрямки для подальшого вдосконалення програми та управління її ризиками та можливостями.

| | *Корисно (для досягнення цілі)* | *Шкодить* |
|---|---|---|
| *m pi* | ***Strengths*** | ***Weaknesses*** |

| | | |
|---|---|---|
| | 1. Простота використання: Програма має простий і зрозумілий інтерфейс для користувача, що полегшує роботу з нею.<br>2. Візуалізація результатів: Можливість візуалізації розташування деталей на полотні у формі графічного представлення на canvas.<br>3. Адаптованість для різних деталей: Програма дозволяє вводити параметри для двох різних типів деталей, що розширює її функціональність.<br>4. | 1. Брак оптимізації алгоритму упаковки: Алгоритм упаковки є простим і може не бути оптимальним у випадках, коли потрібна складна оптимізація розташування деталей.<br>2. Безпека вводу: Недостатня обробка помилок та безпеки вводу користувача. Потребує додаткових заходів безпеки.<br>3. Обмежена інтерактивність: Немає можливості редагування або видалення деталей після розрахунку, що може бути не зручним для користувачів. |
| *Зовнішні* | ***Opportunities*** | ***Threats*** |
| | 1. Оптимізація алгоритму: Заміна алгоритму упаковки на більш оптимальний, що забезпечить кращий розподіл деталей на полотні.<br>2. Розширення функціоналу: Додавання можливості динамічного додавання та видалення деталей, а також збереження конфігурацій для майбутніх переглядів.<br>3. Інтеграція з іншими технологіями: Можливість інтеграції з іншими технологіями для використання розрахунків в інших проектах. | 1. Зміни в технологіях: Швидкі зміни в технологічному ландшафті можуть вимагати переробки програми для відповідності останнім стандартам.<br>2. Проблеми безпеки: Потенційні безпекові загрози, такі як атаки на введення користувача чи витік конфіденційної інформації. |

Рисунок 9 – SWOT-аналіз

*2. Собівартість*

Собівартість визначає вартість виробництва або надання послуги та дозволяє управлінцям приймати обґрунтовані рішення щодо ціноутворення, оптимізації виробництва та маркетингових стратегій.

Обчислення загальних витрат на виробництво дозволяє встановити ціну продажу та визначити рентабельність. Мета створення програмного коду - деталізація витрат для ефективного управління бізнесом та прийняття обгрунтованих рішень.

```
<html>
<head>
    <title>Розрахунок собівартості сумок</title>
</head>
<body>
    <h1>Розрахунок собівартості сумок</h1>

    <form id="costCalculator">
        <h2>Введіть дані для розрахунку собівартості:</h2>

        <label for="modelName">Назва моделі сумки:</label>
        <input type="text" id="modelName" required>

        <label for="totalMaterialCost">Загальна вартість матеріалів (грн):</label>
        <input type="number" id="totalMaterialCost" required>

        <label for="totalLaborCost">Загальна вартість праці (грн):</label>
        <input type="number" id="totalLaborCost" required>

        <label for="totalOverheadCost">Загальні витрати на загальний утримання (грн):</label>
        <input type="number" id="totalOverheadCost" required>

        <label for="quantity">Кількість сумок в партії:</label>
        <input type="number" id="quantity" required>

        <button type="button" id="calculateButton">Розрахувати собівартість</button>
    </form>

    <div id="result">
        <h2>Результати розрахунку:</h2>
        <p  id="modelResult">Собівартість  моделі  <span  id="modelNameResult"></span>:  <span
id="totalCostResult"></span> грн</p>
        <p id="batchCostResult">Витрати на всю партію: <span id="batchTotalCostResult"></span>
грн</p>
    </div>

    <script>
        document.addEventListener("DOMContentLoaded", function() {
            const costCalculator = document.getElementById("costCalculator");
            const calculateButton = document.getElementById("calculateButton");
            const modelNameResult = document.getElementById("modelNameResult");
            const totalCostResult = document.getElementById("totalCostResult");
            const batchTotalCostResult = document.getElementById("batchTotalCostResult");

            calculateButton.addEventListener("click", function() {
                const modelName = document.getElementById("modelName").value;
                const                              totalMaterialCost                              =
parseFloat(document.getElementById("totalMaterialCost").value);
                const                               totalLaborCost                                =
parseFloat(document.getElementById("totalLaborCost").value);
                const                              totalOverheadCost                              =
parseFloat(document.getElementById("totalOverheadCost").value);
                const quantity = parseFloat(document.getElementById("quantity").value);

                // Розрахунок собівартості для однієї сумки
                const unitCost = (totalMaterialCost + totalLaborCost + totalOverheadCost) /
quantity;

                // Розрахунок витрат на всю партію
                const batchTotalCost = unitCost * quantity;

                modelNameResult.textContent = modelName;
                totalCostResult.textContent = unitCost + " грн";
                batchTotalCostResult.textContent = batchTotalCost + " грн";
            });
        });
    </script>
</body>
</html>
```

Для розрахунку собівартості можна використовувати дані про вартість матеріалів, праці та інших витрат. Вище наведено приклад простого коду для розрахунку собівартості на основі цих факторів

Звісно, цей код є простим прикладом і може бути вдосконалений залежно від конкретної бізнес-логіки та вимог. Розрахунок собівартості може включати багато інших факторів, таких як амортизація обладнання, витрати на доставку тощо. Ось деякі ідеї для поліпшення і розширення програми:

Таблиця 4 – Можливості поліпшення

| № | Ідея | Опис |
|---|---|---|
| 1 | Локалізація | Додавання підтримки різних мов для зручності користувачів. |
| 2 | Графіки і діаграми | Можливість візуалізувати результати розрахунків за допомогою графіків чи діаграм для кращого розуміння. |
| 3 | Зберігання даних | Можливість зберігати введені дані та результати розрахунків на стороні користувача або на сервері для подальшого використання. |
| 4 | Експорт даних | Додавання можливості експорту результатів у файли (наприклад, CSV або PDF) для зручності збереження і обміну даними. |
| 5 | Покращений дизайн і стилізація | Поліпшення вигляду та дизайну програми для кращого користувацького досвіду. |
| 6 | Валідація даних | Додавання більш докладної валідації введених даних для уникнення недійсних значень та помилок. |
| 7 | Історія розрахунків | Зберігання історії попередніх розрахунків для можливості перегляду та порівняння результатів. |
| 8 | Підтримка різних одиниць вимірювання | Додавання можливості вибору одиниць вимірювання для введених даних (наприклад, метричні або імперські одиниці). |
| 9 | Автентифікація і облікові записи | Додавання автентифікації для користувачів та можливості створення облікових записів для збереження та обміну даними. |
| 10 | Можливість редагування і видалення | Додавання можливості редагування та видалення раніше введених даних для коригування та поновлення інформації. |

*3. Витрат матеріалів*

```
<html>
<head>
    <title>Розрахунок собівартості сумок</title>
</head>
<body>
    <h1>Розрахунок собівартості сумок</h1>

    <form id="costCalculator">
        <h2>Введіть дані для розрахунку собівартості:</h2>

        <label for="modelName">Назва моделі сумки:</label>
        <input type="text" id="modelName" required>

        <h3>Витрати на матеріали:</h3>

        <label for="material1Name">Тип матеріалу 1:</label>
```

```
        <input type="text" id="material1Name" required>
        <label for="material1Cost">Вартість матеріалу 1 (грн/шт):</label>
        <input type="number" id="material1Cost" required>
        <label for="material1Quantity">Кількість матеріалу 1:</label>
        <input type="number" id="material1Quantity" required>

        <label for="material2Name">Тип матеріалу 2:</label>
        <input type="text" id="material2Name" required>
        <label for="material2Cost">Вартість матеріалу 2 (грн/шт):</label>
        <input type="number" id="material2Cost" required>
        <label for="material2Quantity">Кількість матеріалу 2:</label>
        <input type="number" id="material2Quantity" required>

        <label for="material3Name">Тип матеріалу 3:</label>
        <input type="text" id="material3Name" required>
        <label for="material3Cost">Вартість матеріалу 3 (грн/шт):</label>
        <input type="number" id="material3Cost" required>
        <label for="material3Quantity">Кількість матеріалу 3:</label>
        <input type="number" id="material3Quantity" required>

        <label for="material4Name">Тип матеріалу 4:</label>
        <input type="text" id="material4Name" required>
        <label for="material4Cost">Вартість матеріалу 4 (грн/шт):</label>
        <input type="number" id="material4Cost" required>
        <label for "material4Quantity">Кількість матеріалу 4:</label>
        <input type="number" id="material4Quantity" required>

        <label for="material5Name">Тип матеріалу 5:</label>
        <input type="text" id="material5Name" required>
        <label for="material5Cost">Вартість матеріалу 5 (грн/шт):</label>
        <input type="number" id="material5Cost" required>
        <label for="material5Quantity">Кількість матеріалу 5:</label>
        <input type="number" id="material5Quantity" required>

        <label for="material6Name">Тип матеріалу 6:</label>
        <input type="text" id="material6Name" required>
        <label for="material6Cost">Вартість матеріалу 6 (грн/шт):</label>
        <input type="number" id="material6Cost" required>
        <label for="material6Quantity">Кількість матеріалу 6:</label>
        <input type="number" id="material6Quantity" required>

        <h3>Витрати на працю:</h3>

        <label for="hoursWorked">Кількість годин праці:</label>
        <input type="number" id="hoursWorked" required>

        <label for="hourlyRate">Погодинна ставка (грн/год):</label>
        <input type="number" id="hourlyRate" required>

        <label for="taxRate">Податок (%):</label>
        <input type="number" id="taxRate" required>

        <label for="bonus">Додаткова премія (грн):</label>
        <input type="number" id="bonus" required>

        <button type="button" id="calculateButton">Розрахувати собівартість</button>
    </form>

    <div id="result">
        <h2>Результати розрахунку:</h2>
        <p id="modelResult">Собівартість моделі <span id="modelNameResult"></span>: <span
id="totalCostResult"></span> грн</p>
        <p id="materialCostResult">Витрати на матеріали: <span id="materialCostValue"></span>
грн</p>
        <p id="laborCostResult">Витрати на працю: <span id="laborCostValue"></span> грн</p>
    </div>

    <script>
        document.addEventListener("DOMContentLoaded", function() {
```

```
            const costCalculator = document.getElementById("costCalculator");
            const calculateButton = document.getElementById("calculateButton");
            const modelNameResult = document.getElementById("modelNameResult");
            const totalCostResult = document.getElementById("totalCostResult");
            const materialCostValue = document.getElementById("materialCostValue");
            const laborCostValue = document.getElementById("laborCostValue");

            calculateButton.addEventListener("click", function() {
                const modelName = document.getElementById("modelName").value;

                // Розрахунок витрат на матеріали
                const material1Cost = parseFloat(document.getElementById("material1Cost").value);
                const material1Quantity = parseFloat(document.getElementById("material1Quantity").value);
                const material2Cost = parseFloat(document.getElementById("material2Cost").value);
                const material2Quantity = parseFloat(document.getElementById("material2Quantity").value);
                const material3Cost = parseFloat(document.getElementById("material3Cost").value);
                const material3Quantity = parseFloat(document.getElementById("material3Quantity").value);
                const material4Cost = parseFloat(document.getElementById("material4Cost").value);
                const material4Quantity = parseFloat(document.getElementById("material4Quantity").value);
                const material5Cost = parseFloat(document.getElementById("material5Cost").value);
                const material5Quantity = parseFloat(document.getElementById("material5Quantity").value);
                const material6Cost = parseFloat(document.getElementById("material6Cost").value);
                const material6Quantity = parseFloat(document.getElementById("material6Quantity").value);

                const materialCostTotal = (material1Cost * material1Quantity) + (material2Cost * material2Quantity) +
                    (material3Cost * material3Quantity) + (material4Cost * material4Quantity) +
                    (material5Cost * material5Quantity) + (material6Cost * material6Quantity);

                // Розрахунок витрат на працю
                const hoursWorked = parseFloat(document.getElementById("hoursWorked").value);
                const hourlyRate = parseFloat(document.getElementById("hourlyRate").value);
                const taxRate = parseFloat(document.getElementById("taxRate").value);
                const bonus = parseFloat(document.getElementById("bonus").value);

                const laborCostTotal = (hoursWorked * hourlyRate * (1 + taxRate / 100)) + bonus;

                // Розрахунок загальної собівартості
                const totalCost = materialCostTotal + laborCostTotal;

                modelNameResult.textContent = modelName;
                totalCostResult.textContent = totalCost + " грн";
                materialCostValue.textContent = materialCostTotal + " грн";
                laborCostValue.textContent = laborCostTotal + " грн";
            });
        });
    </script>
</body>
</html>
```

Остання програма для розрахунку собівартості має функціонал та працює належним чином, але є деякі аспекти, які можуть бути вдосконалені:

1. Відсутність валідації введених даних: Поточна програма не має валідації для введених даних, що може призвести до помилок, якщо користувач введе некоректні значення. Варто додати валідацію та повідомлення про помилки.
2. Відсутність можливості зберегти та переглянути попередні розрахунки: Програма не зберігає попередні розрахунки або не надає користувачеві можливість переглянути їх пізніше. Додавання функціоналу для збереження та перегляду історії розрахунків може бути корисним.
3. Відсутність можливості видалення або редагування введених даних: Після введення даних не можна їх видалити або відредагувати. Додавання функціоналу для редагування та видалення записів може поліпшити користувацький досвід.
4. Необов'язкові поля: В програмі всі поля є обов'язковими для заповнення. Це може бути незручно, оскільки користувач може не бажати вводити всі дані. Роблення деяких полів необов'язковими, а інші - обов'язковими, залежно від потреби, може поліпшити програму.
5. Невеликий обсяг введених матеріалів: В програмі введено лише 6 типів матеріалів. Якщо є більше матеріалів, можливо, слід розглянути можливість розширення списку доступних матеріалів.

Загалом, це базовий функціонал для розрахунку собівартості, і його можна вдосконалити в залежності від конкретних потреб і вимог користувачів.

Нижче наведена програма, яка об'єднує різні функції, включаючи схеми розкрою, собівартість, витрати матеріалу, керування запасами, розрахунок собівартості та планування виробництва:

### 3.3. Експерименти та тестування розробленого програмного продукту

При розгляді програмних продуктів важливо враховувати їхні переваги і недоліки. Ось деякі можливі недоліки в програмному продукті:

1. Потреба в Інтернет-підключенні: Програмні продукти можуть вимагати постійного Інтернет-підключення для роботи, що може бути незручним, особливо в умовах обмеженого доступу до мережі.
2. Висока вартість: Окремі програмні продукти можуть бути дорогими для користувачів або підприємств. Вартість ліцензій або підписок може бути значним чинником вибору.
3. Обмежені можливості: Деякі програмні продукти можуть бути обмеженими за функціональністю або можливостями порівняно з конкурентами. Це може призвести до обмеженого використання або необхідності використовувати додаткові програми.
4. Проблеми з безпекою: Програмні продукти можуть містити уразливості, які стають об'єктом атак зловмисників. Постійне оновлення і забезпечення безпеки може бути проблематичним завданням.
5. Сумісність: Проблеми з сумісністю можуть призвести до проблем при інтеграції з існуючими системами.
6. Підтримка та обслуговування: Важливо мати доступ до якісної та своєчасної підтримки та обслуговування. Нестача підтримки може призвести до затримок у вирішенні проблем або усуненні недоліків.
7. Інтерфейс користувача: Неінтуїтивний або складний інтерфейс користувача може ускладнити використання програмного продукту і зменшити продуктивність користувачів.
8. Приватність і збереження даних: Питання щодо приватності та збереження даних можуть бути серйозними, особливо в продуктах, які обробляють чутливі або особисті інформацію.
9. Відсутність оновлень: Відсутність регулярних оновлень може призвести до вразливостей та невідповідності змінюючимся потребам користувачів.
10. Залежність від розробника: Якщо продукт розробляється невеликим розробником або компанією, то існує ризик, що вони можуть припинити розробку або підтримку продукту, що може залишити користувачів без необхідної допомоги.

Важливо розглядати ці недоліки, але також брати до уваги переваги та унікальність кожного програмного продукту. Оцінка відповідності ваших конкретних потреб і вимог допоможе визначити, чи програмний продукт підходить для вас чи вашої компанії.

В фінальний код було поміщено усі необхідні функції яка може виконувати програма. І він має наступний вигляд:

```
    <html lang="en">
    <head>
        <meta charset="UTF-8">
        <meta  name="viewport"  content="width=device-width,  initial-
scale=1.0">
        <title>Схема розміщення деталей на полотні</title>
        <style>
            body {
                font-family: 'Arial', sans-serif;
                max-width: 800px;
                margin: 0 auto;
                padding: 20px;
                background-color: #f4f4f4;
            }

            h1, h2, h3 {
                color: #333;
            }

            form {
                background-color: #fff;
                padding: 20px;
                margin-bottom: 20px;
                border-radius: 8px;
                box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
            }

            label {
                display: block;
                margin-bottom: 5px;
                color: #555;
            }

            input {
                width: 100%;
                padding: 8px;
                margin-bottom: 10px;
                box-sizing: border-box;
                border: 1px solid #ccc;
                border-radius: 4px;
                font-size: 14px;
```

```
            }

            button {
                background-color: #4caf50;
                color: #fff;
                border: none;
                padding: 10px 20px;
                border-radius: 4px;
                cursor: pointer;
                font-size: 16px;
            }

            #canvas {
                border: 1px solid blue;
                position: relative;
                margin-top: 20px;
                height: 300px;
                background-color: #f0f0f0;
            }

            .detail {
                position: absolute;
                border: 1px solid red;
                background-color: #ff8080;
            }

            #result {
                background-color: #fff;
                padding: 20px;
                border-radius: 8px;
                box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
            }
        </style>
    </head>
    <body>
        <h1>Схема розміщення деталей на полотні</h1>

        <form id="layoutForm">
            <h2>Полотно</h2>
            <label for="canvasWidth">Ширина полотна (дм):</label>
            <input type="number" id="canvasWidth" required>

            <label for="canvasHeight">Висота полотна (дм):</label>
            <input type="number" id="canvasHeight" required>

            <h2>Деталь №1</h2>
            <label for="detail1Width">Ширина деталі №1 (дм):</label>
            <input type="number" id="detail1Width" required>

            <label for="detail1Height">Висота деталі №1 (дм):</label>
            <input type="number" id="detail1Height" required>
```

```
            <label for="detail1Count">Кількість деталей №1:</label>
            <input type="number" id="detail1Count" required>

            <h2>Деталь №2</h2>
            <label for="detail2Width">Ширина деталі №2 (дм):</label>
            <input type="number" id="detail2Width" required>

            <label for="detail2Height">Висота деталі №2 (дм):</label>
            <input type="number" id="detail2Height" required>

            <label for="detail2Count">Кількість деталей №2:</label>
            <input type="number" id="detail2Count" required>

            <button                                          type="button"
id="calculateLayoutButton">Розрахувати розташування</button>
        </form>

        <form id="costCalculatorForm">
            <h2>Дані для розрахунку собівартості:</h2>
            <label for="modelName">Назва моделі сумки:</label>
            <input type="text" id="modelName" required>

            <label      for="totalMaterialCost">Загальна      вартість
матеріалів (грн):</label>
            <input type="number" id="totalMaterialCost" required>

            <label   for="totalLaborCost">Загальна   вартість   праці
(грн):</label>
            <input type="number" id="totalLaborCost" required>

            <label    for="totalOverheadCost">Загальні    витрати    на
загальний утримання (грн):</label>
            <input type="number" id="totalOverheadCost" required>

            <label for="quantity">Кількість сумок в партії:</label>
            <input type="number" id="quantity" required>

            <button                                          type="button"
id="calculateCostButton">Розрахувати собівартість</button>
        </form>

        <div id="canvas"></div>

        <div id="result">
            <h2>Результати розрахунку:</h2>
            <p     id="modelResult">Собівартість     моделі     <span
id="modelNameResult"></span>: <span id="totalCostResult"></span> </p>
            <p  id="batchCostResult">Витрати  на  всю  партію:  <span
id="batchTotalCostResult"></span> </p>
            <p  id="materialCostResult">Витрати  на  матеріали:  <span
```

```
id="materialCostValue"></span> </p>
            <p    id="laborCostResult">Витрати    на    працю:    <span
id="laborCostValue"></span> </p>
        </div>

        <script>
            document.addEventListener("DOMContentLoaded",   function()
{
                const                   layoutForm                   =
document.getElementById("layoutForm");
                const               costCalculatorForm               =
document.getElementById("costCalculatorForm");
                const             calculateLayoutButton              =
document.getElementById("calculateLayoutButton");
                const              calculateCostButton               =
document.getElementById("calculateCostButton");
                const                modelNameResult                 =
document.getElementById("modelNameResult");
                const                totalCostResult                 =
document.getElementById("totalCostResult");
                const               materialCostValue                =
document.getElementById("materialCostValue");
                const                laborCostValue                 =
document.getElementById("laborCostValue");

                calculateLayoutButton.addEventListener("click",
calculateLayout);
                calculateCostButton.addEventListener("click",
calculateCost);

                function calculateLayout() {
                    const                canvasWidth                =
parseFloat(document.getElementById("canvasWidth").value);
                    const                canvasHeight               =
parseFloat(document.getElementById("canvasHeight").value);

                    const                detail1Width               =
parseFloat(document.getElementById("detail1Width").value);
                    const               detail1Height               =
parseFloat(document.getElementById("detail1Height").value);
                    const                detail1Count               =
parseInt(document.getElementById("detail1Count").value);

                    const                detail2Width               =
parseFloat(document.getElementById("detail2Width").value);
                    const               detail2Height               =
parseFloat(document.getElementById("detail2Height").value);
                    const                detail2Count               =
parseInt(document.getElementById("detail2Count").value);

                    const canvas = document.getElementById("canvas");
```

```
                    canvas.style.width = canvasWidth + "cm";
                    canvas.style.height = canvasHeight + "cm";
                    canvas.innerHTML = "";

                    const details = [];
                    for (let i = 0; i < detail1Count; i++) {
                        details.push({  width:  detail1Width,  height:
detail1Height });
                    }
                    for (let i = 0; i < detail2Count; i++) {
                        details.push({  width:  detail2Width,  height:
detail2Height });
                    }

                    const  packer  =  new  RectanglePacker(canvasWidth,
canvasHeight);
                    const packedRectangles = packer.pack(details);

                    for (const rect of packedRectangles) {
                        const detail = document.createElement("div");
                        detail.className = "detail";
                        detail.style.width = rect.width + "cm";
                        detail.style.height = rect.height + "cm";
                        detail.style.left = rect.x + "cm";
                        detail.style.top = rect.y + "cm";
                        canvas.appendChild(detail);
                    }
                }

                function calculateCost() {
                    const                  modelName                =
document.getElementById("modelName").value;
                    const             totalMaterialCost             =
parseFloat(document.getElementById("totalMaterialCost").value);
                    const               totalLaborCost              =
parseFloat(document.getElementById("totalLaborCost").value);
                    const             totalOverheadCost             =
parseFloat(document.getElementById("totalOverheadCost").value);
                    const                  quantity                 =
parseFloat(document.getElementById("quantity").value);

                    const    unitCost    =    (totalMaterialCost    +
totalLaborCost + totalOverheadCost) / quantity;
                    const batchTotalCost = unitCost * quantity;

                    modelNameResult.textContent = modelName;
                    totalCostResult.textContent = unitCost + " грн";
                    batchTotalCostResult.textContent  =  batchTotalCost
+ " грн";
                    materialCostValue.textContent  =  totalMaterialCost
+ " грн";
```

```
                    laborCostValue.textContent = totalLaborCost + "
грн";
                }

                class RectanglePacker {
                    constructor(width, height) {
                        this.width = width;
                        this.height = height;
                        this.rectangles = [{ x: 0, y: 0, width,
height }];
                    }

                    pack(rectangles) {
                        const result = [];

                        for (const rect of rectangles) {
                            for (const bin of this.rectangles) {
                                if (bin.width >= rect.width &&
bin.height >= rect.height) {
                                    result.push({ x: bin.x, y: bin.y,
width: rect.width, height: rect.height });

                                    if (bin.width > rect.width) {
                                        this.rectangles.push({ x:
bin.x + rect.width, y: bin.y, width: bin.width - rect.width, height:
rect.height });
                                    }
                                    if (bin.height > rect.height) {
                                        this.rectangles.push({ x:
bin.x, y: bin.y + rect.height, width: bin.width, height: bin.height -
rect.height });
                                    }


this.rectangles.splice(this.rectangles.indexOf(bin), 1);
                                    break;
                                }
                            }
                        }

                        return result;
                    }
                }
            });
        </script>
    </body>
</html>
```

Запустивши програму користувач побачить наступний інтерфейс:

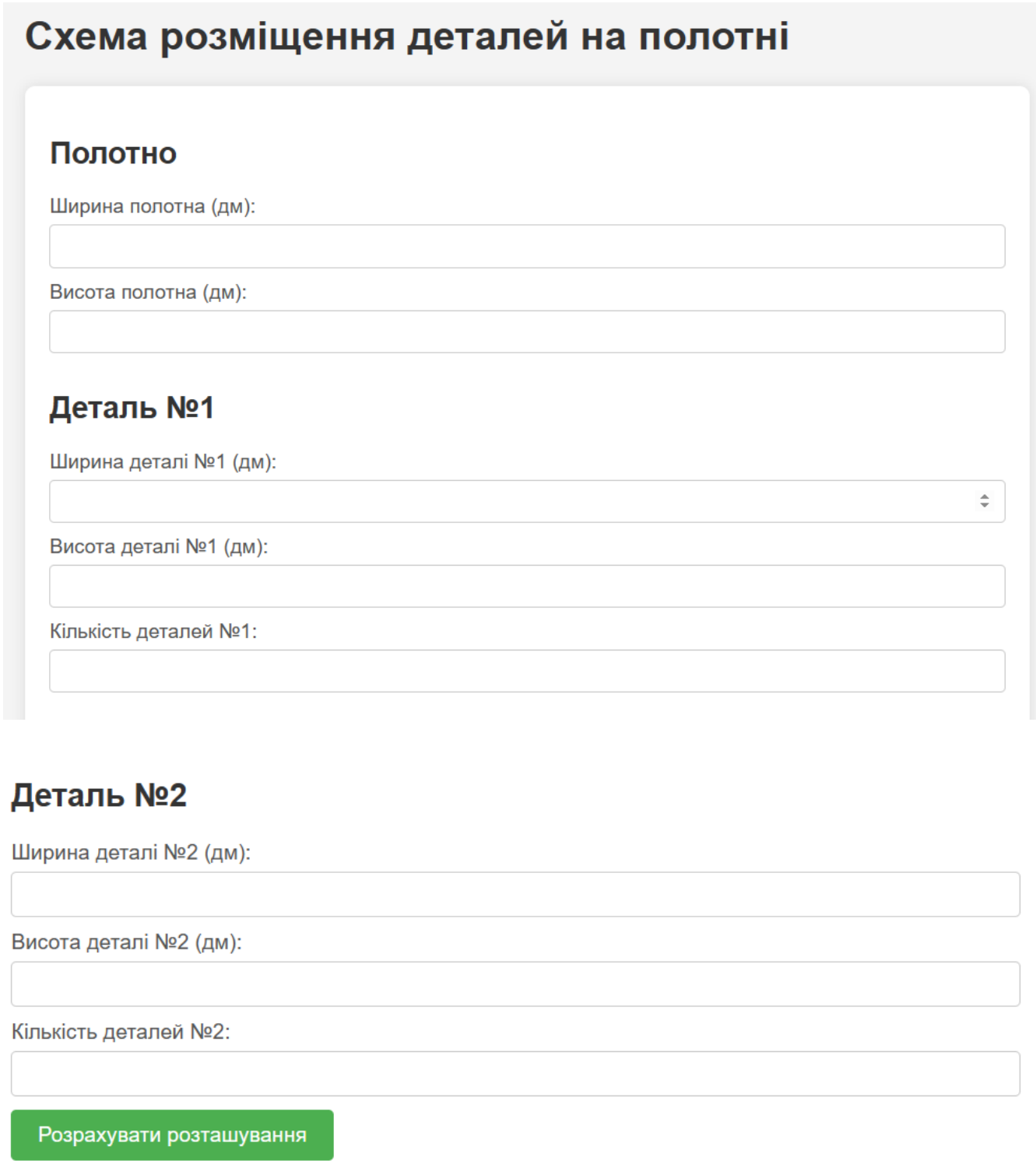
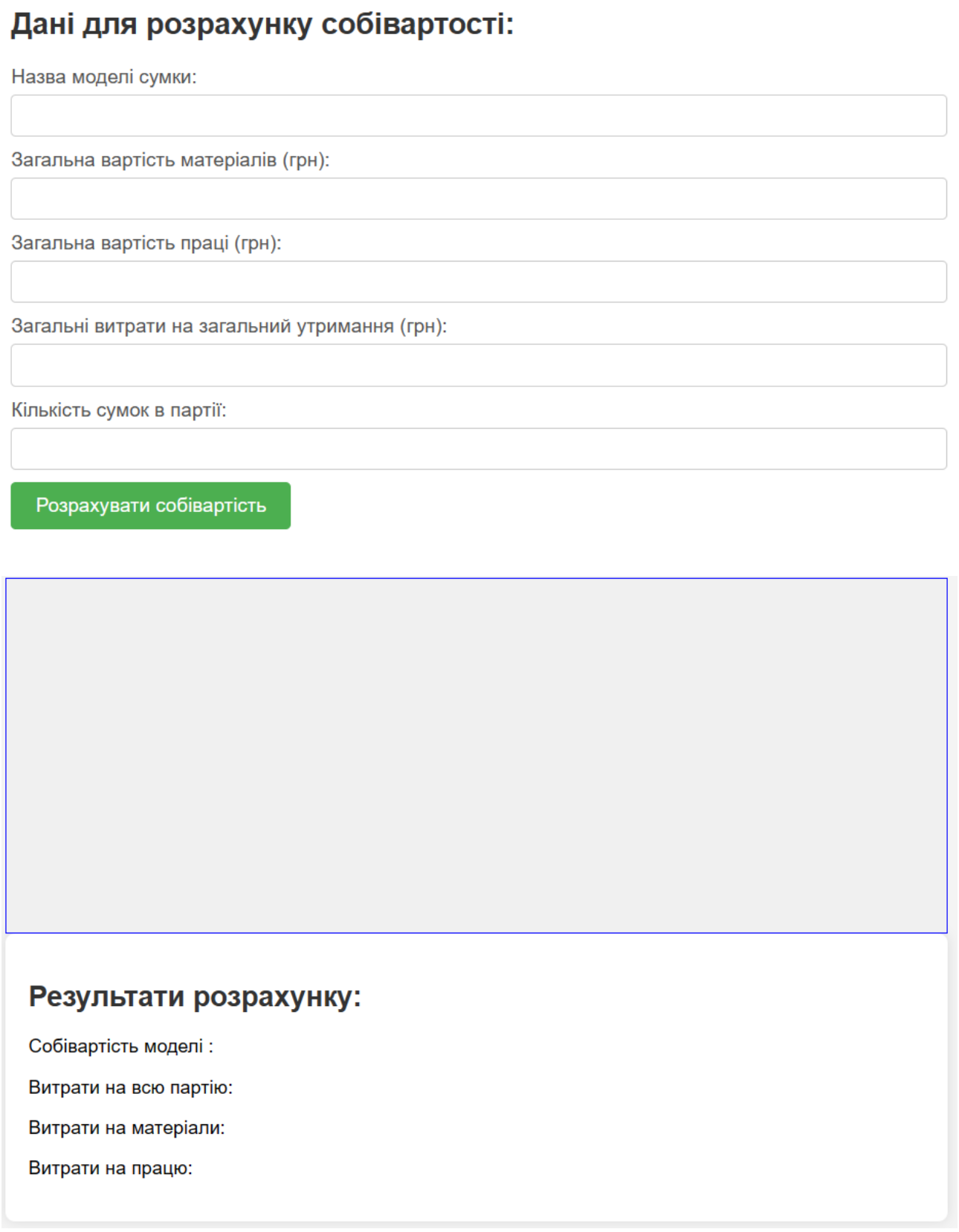



Рисунок 10 – Розроблена сторінка схеми розміщення деталей на полотні




Продовження рисунку 10 – Розроблена сторінка схеми розміщення деталей на полотні.

Код має вигляд веб-сторінки з HTML, CSS і JavaScript кодом, що використовується для розрахунку розміщення деталей на полотні та для розрахунку собівартості виготовлення сумок. Ось деякі ключові аспекти цього коду:

1. HTML розмітка:

- Сторінка має дві форми - одна для розрахунку розміщення деталей на полотні, а інша для розрахунку собівартості виготовлення сумок.
- Є також блок  для відображення результатів розрахунків та <canvas> для відображення схеми розміщення деталей.

2. CSS стилі:

- Стилі визначають зовнішній вигляд сторінки, форм, кнопок і результатів розрахунків.
- Використовуються різні кольори та стилі для виділення елементів.

3. JavaScript код:

- Є дві основні функції - calculateLayout для розрахунку розміщення деталей на полотні та calculateCost для розрахунку собівартості виготовлення сумок.
- Для розміщення деталей використовується алгоритм RectanglePacker, який намагається ефективно впакувати прямокутники на полотно.

Цей код може бути корисним для розробки інтерактивних інструментів для дизайнерів або виробників сумок, де важливо розраховувати оптимальне розташування деталей та визначати собівартість виробництва. Також ця програма об'єднує в собі введення користувача, алгоритми розрахунків та візуалізацію результатів, що може бути корисним для тих, хто працює у галузі розробки та виробництва.

Головними елементами цієї програми є:

1. HTML Розмітка:

- <form> для розміщення деталей на полотні (з ідентифікатором "layoutForm").
- Є дві групи полів для введення даних про полотно та дві різні деталі.

- <form> для розрахунку собівартості виготовлення сумок (з ідентифікатором "costCalculatorForm").
- Поля для введення назви моделі, вартості матеріалів, праці, загальних витрат та кількості одиниць.

2. CSS Стилі:

- Стилізація для зовнішнього вигляду елементів сторінки, таких як форми, кнопки, блоки результатів та полотно для розташування деталей.

3. JavaScript Код:

- Доданий обробник подій DOMContentLoaded, що викликається, коли весь HTML завантажено і оброблено.
- Є дві основні функції:
  - calculateLayout: Обчислює розташування деталей на полотні за допомогою алгоритму RectanglePacker та оновлює відповідне полотно на сторінці.
  - calculateCost: Обчислює собівартість виготовлення сумок на основі введених даних та виводить результати на сторінку.

4. RectanglePacker Клас:

- Клас, який відповідає за алгоритм ефективного розташування прямокутних деталей на полотні.
- Містить метод pack, який намагається впакувати прямокутники в доступні регіони на полотні.

5. HTML Канвас та Деталі:

- Елемент  з ідентифікатором "canvas" використовується для відображення полотна.
- Деталі, розміщені на полотні, представлені як елементи  з класом "detail", встановленим розміром та позицією відносно полотна.

6. Результати Розрахунків:

- Блок  з ідентифікатором "result" використовується для відображення результатів розрахунків.

- Різні <p> елементи виводять різні аспекти результатів, такі як назва моделі, собівартість, витрати на матеріали та працю.

Загалом, код представляє собою сторінку з двома формами для введення параметрів, блоками для відображення полотна та результатів, а також вбудованим JavaScript-кодом для розрахунків та маніпуляцій з DOM.

Варто деталізувати код в розрізі окремих частин.

1. HTML частина:

<html lang="en">: Визначає мову документа як англійську.

<head>: Містить мета-інформацію та посилання на стилі та скрипти.

<meta charset="UTF-8">: Вказує кодування документа як UTF-8.

<meta name="viewport" content="width=device-width, initial-scale=1.0">: Налаштування для адаптивності на різних пристроях.

<title>Схема розміщення деталей на полотні</title>: Заголовок веб-сторінки.

<style>: Включає стилі CSS для оформлення сторінки.

2. CSS-стилі (описують зовнішній вигляд елементів сторінки):

body: Задає шрифт, ширину та колір тла сторінки.

form: Стилі для форм.

label, input, button: Стилі для міток, полів вводу та кнопок.

#canvas: Стилі для області малювання (можливо, для відображення деталей).

.detail: Стилі для окремих деталей.

#result: Стилі для відображення результатів розрахунку.

3. JavaScript-частина:

document.addEventListener("DOMContentLoaded", function() {...});: Обробник події, який виконується після завантаження DOM.

Константи для елементів форм, кнопок та результатів.

calculateLayout та calculateCost: Функції для розрахунку розташування деталей та вартості виробництва.

RectanglePacker: Клас, який відповідає за упаковку деталей на полотні.

4. HTML-елементи та форми:

Два блоки форм (layoutForm та costCalculatorForm), кожен з яких містить відповідні поля для введення даних.

canvas: Порожній блок, який може використовуватися для відображення малюнків або деталей.

result: Блок для виведення результатів розрахунку.

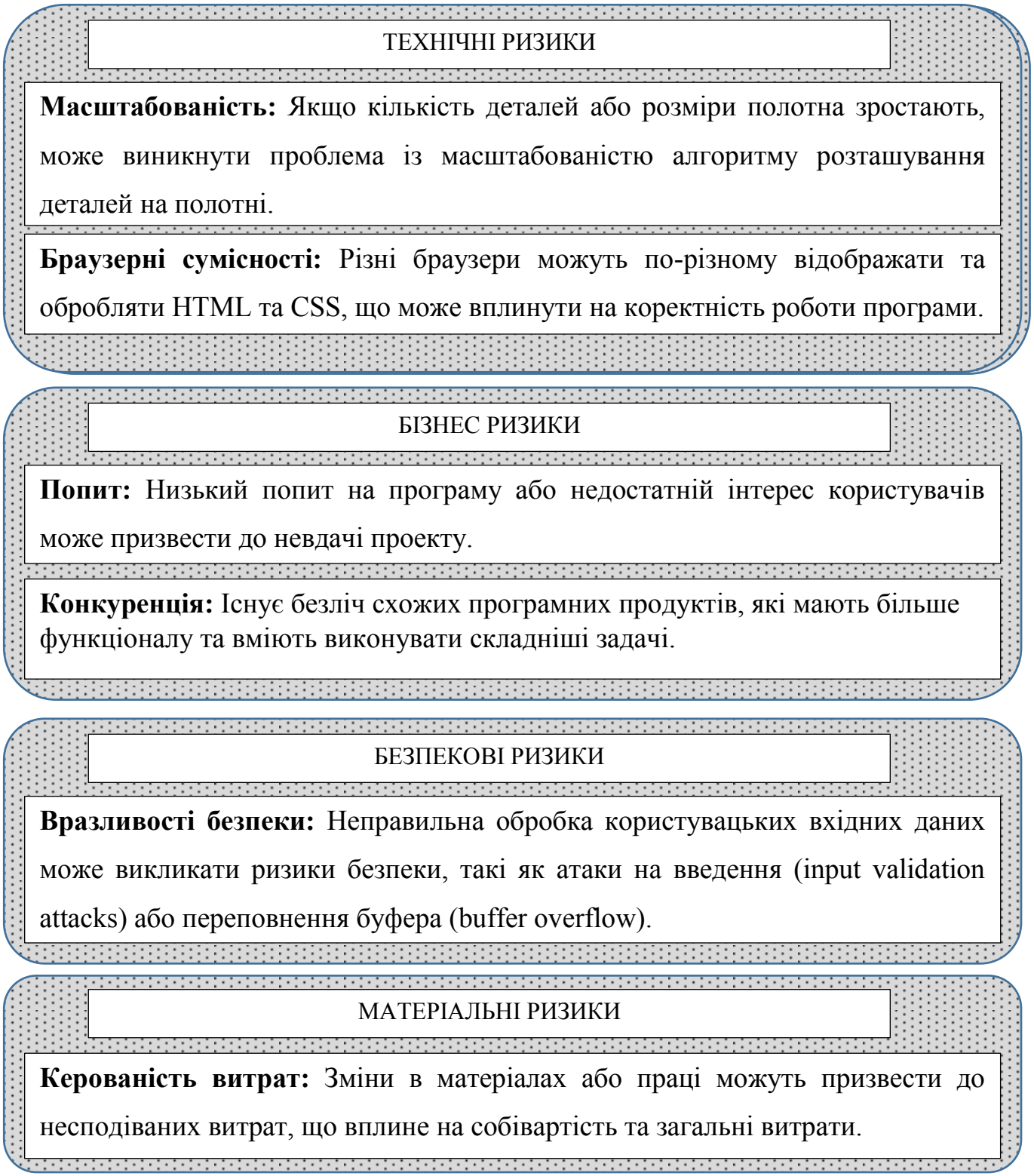



Рисунок 11 – Risk Analysis (Аналіз ризиків)

**Функціональність**

Розміщення деталей - *Оцінка:* Порівняти можливості розміщення деталей на полотні із сучасними графічними редакторами, такими як "AutoCAD", "Inkscape", зокрема, з урахуванням введених розмірів і кількості деталей.

Візуалізація - *Оцінка:* Порівняти способи візуалізації розміщення деталей, зокрема, порівняти зручність сприйняття відображення на полотні.

**Використання ресурсів**

Час розрахунку – *Вимір:* Визначити час, необхідний для розрахунку розміщення деталей на полотні в залежності від їхньої кількості та розмірів. *Порівняння:* Порівняти цей час із аналогічними програмами для визначення швидкодії.

Використання пам'яті – *Вимір:* Вивчити обсяг оперативної пам'яті, який використовує програма при розрахунках. *Порівняння:* Порівняти використання пам'яті із схожими інструментами.

**Користувацький інтерфейс**

Дизайн та ергономіка – *Оцінка:* Розглянути дизайн інтерфейсу та його відповідність стандартам UX/UI. *Порівняння:* Порівняти з іншими графічними редакторами.

Зручність використання – *Оцінка:* Проаналізувати легкість використання та інтуїтивність інтерфейсу. *Порівняння:* Порівняти рівень зручності із конкурентами.

Рисунок 12 – Benchmarking (Порівняння)

Також необхідно розглянути аналіз ризиків (Risk Analysis) та порівняння (Benchmarking) для наведеного програмного продукту.

Обидва аспекти – ризиковий аналіз та порівняння – є важливими для розуміння сильних і слабких сторін програмного продукту та його конкурентноспроможності на ринку. Під час розробки та після випуску, важливо враховувати знайдені ризики та можливості для постійного вдосконалення продукту.

## Висновки до розділу 3

В цьому розділі ми розглянули процес розробки та дослідження програмного забезпечення для автоматизованої підготовки інформації про деталі шкіргалантереї. Почавши з опису математичних моделей та алгоритмів для розробки програми розкрою деталей шкірогалантереї, ми перейшли до розгляду різних аспектів цього завдання.

У розділі 3.1 було визначено, що основним завданням є розробка програми, яка спрощує процес розкрою деталей для виробництва сумок. Програма має надавати можливість створення схем розкладання деталей, розрахунку собівартості, визначення витрат матеріалів та генерації звітів для керівництва. Ключовими інструментами для розробки цієї програми ми обрали HTML та JavaScript.

У розділі 3.2 було надано стратегію розробки програмного забезпечення на базі обраної технології. Також розглядались можливості розширення функціоналу, покращення інтерфейсу користувача та додавання функцій для кращого використання програми.

В цілому, розділ 3 надав важливий огляд завдань і можливостей, пов'язаних з розробкою програмного забезпечення для автоматизованої підготовки інформації про деталі шкіргалантереї. Детальний аналіз математичних моделей, алгоритмів та технологій надає основу для подальшої розробки програми, яка може сприяти ефективному виробництву та керуванню ресурсами в цій галузі.

# ВИСНОВКИ

У даному дослідженні було ретельно розглянуто проблему автоматизованої підготовки інформації про деталі шкіргалантереї та розробки відповідного програмного продукту.

Усі розділи дослідження спільно сприяли створенню бази для подальшого розвитку автоматизованого рішення для підготовки інформації про деталі шкіргалантереї. Результати цього дослідження можуть бути використані як основа для практичної реалізації та впровадження ефективних інструментів в галузі шкіргалантереї.

Це дослідження відкрило широкий спектр можливостей для впровадження автоматизації в галузі шкіргалантереї. Наша робота обговорює важливість розробки програмних рішень для оптимізації процесів підготовки інформації про деталі, що може сприяти підвищенню ефективності та конкурентоспроможності підприємств у цій галузі.

Огляд існуючих систем та програм для обробки інформації про деталі шкіргалантереї показав, що існують певні особливості в існуючих підходах. Наша робота наголошує на необхідності розвитку більш ефективних і інноваційних рішень, які можуть враховувати поточні вимоги галузі.

В процесі написання магістерської роботи було створено програму що об'єднує розташування деталей на полотні та розрахунок собівартості в єдиному інтерфейсі. Вона може бути корисною для дизайнерів, які працюють над розміщенням деталей для виробництва сумок, а також для управлінців, які хочуть швидко розрахувати собівартість своєї продукції. Алгоритм упаковки прямокутників допомагає ефективно використовувати полотно та мінімізувати втрати матеріалу.

Програма використовує алгоритм упаковки прямокутників для оптимального розташування деталей на полотні, що дозволяє максимізувати використання матеріалу та зменшити втрати.

Візуалізація розташування деталей на полотні здійснюється за допомогою прямокутників, що дозволяє користувачеві легко спостерігати за результатами.

Користувач може введенням даних швидко розрахувати собівартість виробу та витрати на виробництво всієї партії. Результати розрахунку виводяться у зрозумілій формі, надаючи зрозумілу інформацію про витрати на матеріали, працю та загальні витрати.

Інтерфейс програми створений для зручності користувача, з використанням форм для введення даних та виділення різних розділів, щоб полегшити організацію інформації.

Ця програма може бути важливим інструментом для підприємців і дизайнерів, проте вона потребує багатьох допрацювань та уточнень в майбутньому.

# Список використаних джерел


1. Автоматизоване проектування жіночих рукавичок. М. М. Кохтярук. URL: https://er.knutd.edu.ua/bitstream/123456789/13493/1/NRMSE2019_V1_P186-187.pdf

2. Легка промисловість україни: сучасний стан та перспективи розвитку. І. О. Максименко, В. І. Бокій. URL: http://journals.khnu.km.ua/vestnik/pdf/ekon/2009_3_2/pdf/077-080.pdf

3. Конкурентоспроможність вітчизняних шкіргалантерейних виробів. Проблеми галузі та пропозиції щодо їх вирішення. В.В. Філіпенко, Т.Я. Піддубна. URL: https://er.knutd.edu.ua/bitstream/123456789/6319/1/V53_P017-022.pdf

4. Проблеми впровадження автоматизованих систем в легкій промисловості. Н. А. Єфременкова. URL:https://er.knutd.edu.ua/bitstream/123456789/16218/1/Iefremenkova_N._Problems_of_introducing_automated_systems_in_the_light_industry.pdf

5. Підготовка вхідної інформації про схеми суміщення деталей взуття для автоматизованого розкрою. В.І. Чупринка, О.О. Хоменко, М.М. Шкоденко. URL: https://er.knutd.edu.ua/bitstream/123456789/6963/1/V46_P022-028.pdf

6. Textile and Clothing Design Technology. Tom Cassidy, Parikshit Goswami. URL:https://books.google.ca/books?hl=uk&lr=&id=wmpQDwAAQBAJ&oi=fnd&pg=PA375&dq=1.%09Gerber+Technology+AccuMark&ots=zlo-dLO1d7&sig=2wiEEfgUBPGtToQdEAj_E_qQfcA&redir_esc=y#v=onepage&q=1.%09Gerber%20Technology%20AccuMark&f=false

7. AccuMark software. URL: https://www.gerbertechnology.com/landing-pages/accumark-family/

8. OPTITEX. URL: https://optitex.com/

9. TUKATECH. URL: https://tukatech.com/tuka3d/

10. Modaris. URL: https://www.lectra.com/en/products/modaris-expert

11. BROWZWEAR. URL: https://browzwear.com/products/v-stitcher

12. Application of 3D Virtual Prototyping Technology to the Integration of Wearable Antennas into Fashion Garments. URL: https://www.mdpi.com/2227-7080/10/3/62

13. ISO 14001. URL: https://www.iso.org/iso-14001-environmental-management.html

14. ISO 9001. URL: https://www.iso.org/iso-9001-quality-management.html

15. REACH. URL: https://echa.europa.eu/regulations/reach/understanding-reach

16. STANDARD 100 by OEKO‑ TEX. URL: https://www.oeko-tex.com/importedmedia/downloadfiles/STANDARD_100_by_OEKO-TEX_R__-_Standard_en.pdf

17. ASTM Internationa. URL: https://www.astm.org/