**CHAPTER 1**

**INTRODUCTION**

Embark on a transformative journey with our recommended hair care project that delves into the art and science of nurturing and enhancing your locks. This comprehensive venture aims to guide individuals towards healthier, more vibrant hair by exploring a curated selection of proven hair care practices, products, and routines. Our project navigates the landscape of ingredients beneficial for diverse hair types. Dive into the chemistry of shampoos, conditioners, and styling products to make informed choices for your unique hair profile.

* 1. **OVERVIEW**

The Hair Care Recommender project is an innovative endeavor aimed at revolutionizing the way individuals approach hair care product selection and purchasing. At its core, the project seeks to address the diverse and often complex needs of users by providing personalized recommendations tailored to their unique hair types, concerns, preferences, and requirements.

At the heart of the project lies a sophisticated user profile management system. This system empowers users to meticulously create and maintain profiles that capture essential details about their hair, including its type (straight, wavy, curly, etc.), specific concerns (dryness, frizz, damage), preferred ingredients (natural oils, specific additives), and any allergies. These comprehensive profiles serve as the backbone for the recommendation engine, enabling it to generate custom-tailored product suggestions based on each user's unique characteristics and preferences.

Central to the project's functionality is its extensive product database, meticulously curated to encompass a wide range of hair care products, including shampoos, conditioners, serums, masks, and more. Each product within the database is meticulously categorized and tagged with attributes such as ingredients, suitability for different hair types, targeted concerns, and user reviews. This wealth of information allows the recommendation engine to draw upon a rich pool of options when generating personalized recommendations for users.

In addition to its recommendation capabilities, the Hair Care Recommender project boasts robust search and filtering functionality, empowering users to explore the product database with ease. Whether users are searching for a specific product or seeking to narrow down their

options based on criteria such as hair type, concerns, ingredients, brand, or price range, the platform provides intuitive tools to streamline the browsing experience.

User engagement and community interaction are also key focal points of the project. Through features like user reviews and ratings, individuals can share their experiences with different products, providing valuable insights and recommendations to fellow users. This collaborative approach not only enhances the platform's credibility but also fosters a sense of community among users with shared hair care interests and concerns.

Moreover, the project prioritizes user convenience and accessibility through features like shopping cart functionality and purchase history tracking. By seamlessly integrating these features into the platform, users can effortlessly add products to their carts, make purchases directly through the platform, and track their transaction history over time.

Implemented with a combination of cutting-edge technologies and a structured implementation plan, the Hair Care Recommender project aims to deliver a user-centric solution that redefines the hair care shopping experience. By leveraging the power of data-driven recommendations, intuitive user interfaces, and community-driven engagement, the project aspires to empower individuals to make informed decisions and achieve healthier, more beautiful hair with confidence.

* 1. **OBJECTIVE**

The primary objective of the Hair Care Recommender project is to revolutionize the way individuals discover and select hair care products by providing a personalized, data-driven recommendation system. Through the amalgamation of advanced machine learning algorithms and user-centric design principles, the project aims to address the complexities and nuances inherent in hair care product selection. By empowering users to create detailed profiles encompassing their unique hair characteristics, preferences, and concerns, the system endeavors to offer tailored recommendations that align with each user's individual needs and preferences.

Through the seamless integration of user feedback and iterative algorithm refinement, the Hair Care Recommender seeks to continually enhance the accuracy, relevance, and efficacy of its recommendations, thereby fostering a more informed and satisfying hair care journey for consumers. Furthermore, the project aspires to contribute to the advancement of data product management strategies within the beauty industry, serving as a model for leveraging data-driven insights to enhance user experiences, drive customer engagement, and foster brand loyalty.

Ultimately, the overarching goal of the Hair Care Recommender project is to empower individuals to achieve healthier, happier hair by facilitating informed decision-making and personalized product selection in the ever-evolving landscape of hair care.

* 1. **PURPOSE**

In a world where the choices for hair care products are abundant and diverse, finding the perfect match for one's unique hair needs can be daunting. The Hair Care Recommender project aims to simplify this process by offering personalized recommendations tailored to individual hair types, concerns, and preferences. By harnessing the power of data analytics and machine learning, the project seeks to empower users to make informed decisions, enhance their hair care routines, and ultimately, achieve healthier and happier hair. Through this initiative, we aspire to redefine the way people approach hair care product selection, making it more personalized, efficient, and enjoyable.

1. **Enhancing User Experience:** The project aims to improve the user experience by simplifying the process of selecting hair care products. Many individuals struggle to find products that suit their specific hair type, concerns, and preferences. By providing personalized recommendations, the project aims to alleviate this challenge and streamline the product discovery process.
2. **Meeting Individual Needs:** Hair care is not one-size-fits-all. Different individuals have unique hair characteristics, concerns, and preferences. The project seeks to address this diversity by tailoring recommendations to each user's specific needs. Whether someone has curly, straight, oily, or dry hair, the recommender system aims to suggest products that are best suited to their hair type and concerns.
3. **Improving Product Selection:** With a vast array of hair care products available on the market, selecting the right ones can be overwhelming. The project aims to simplify this process by providing users with personalized recommendations based on factors such as ingredients, benefits, and user reviews. By offering relevant suggestions, the project helps users make more informed decisions and find products that align with their preferences and requirements.
4. **Driving Customer Satisfaction and Loyalty:** By assisting users in finding products that work well for them, the project aims to enhance customer satisfaction and foster brand loyalty. When users have positive experiences with recommended products, they are more likely to trust the recommender system and continue using it for future purchases. This, in turn, can lead to increased customer retention and brand advocacy.
5. **Advancing Data Product Management Strategies:** Beyond its immediate benefits to users, the project also serves as a demonstration of effective data product management strategies. By leveraging data analytics, machine learning algorithms, and user feedback, the project showcases how businesses can create valuable data-driven solutions to enhance user experiences and drive business growth in the beauty industry and beyond.

**CHAPTER 2**

**SYSTEM AND SOFTWARE REQUIREMENTS AND SPECIFICATIONS**

The program works on Desktop PC and is executed using a PHP 5 interface which interacts with a MySQL database running on local host.

* 1. **FUNCTIONAL REQUIREMENTS**

A description of the facility or feature required. Functional requirements deal with what the system should do or provide for users. They include description of the required functions, outlines of associated reports or online queries, and details of data to be held in the system.

* + 1. **Interface Requirements:**

1. User Registration/Login: Provide a seamless registration and login process for users to create accounts and access the recommendation system.
2. Profile Creation: Enable users to create detailed profiles by inputting information about their hair type, concerns, preferences, and any specific ingredients they prefer or avoid.
3. Product Search and Recommendations: Implement a user-friendly interface where users can search for hair care products and receive personalized recommendations based on their profile information.
4. Product Details: Display detailed information about recommended products, including ingredients, benefits, user reviews, and ratings.

**2.2 NON-FUNCTIONAL REQUIREMENTS:**

Non-functional requirements define the overall qualities or attributes of the resulting system.

* + 1. **Usability**
* Design an intuitive and accessible user interface with multilingual support for enhanced user experience.
  + 1. **Security**

Security requirements are included in a system to ensure:

* + - * All questions and answers are well secured
      * SQL injection is prevented
      * Implement data encryption, secure authentication, and secure communication to protect user data and prevent unauthorized access.
    1. **Performance**
* Ensure fast response times, scalability to handle increasing user loads, and high reliability with minimal downtime.
  + 1. **Reliability**
* Implement robust error handling, backup, and recovery mechanisms to ensure system stability and data integrity.

**2.3 SOFTWARE REQUIREMENTS**

Software requirements refer to the specifications and functionalities that a software application must possess to meet the needs of its users. These requirements include features, user interface design, and performance, compatibility, and security considerations.

Programming language : Python, MySQL.

Operating system : ANY OS (Recommended**:** Windows8,

Windows Vista, Windows XP) Application required : Standalone desktop application & MySQL Coding language : Python, HTML, CSS, JavaScript

**2.4 HARDWARE REQUIREMENTS**

Hardware requirements pertain to the necessary hardware components and specifications that a computer system must have to effectively run a software application. This includes the processor, memory (RAM), storage (hard drive or SSD), graphics card, and other peripheral devices such as printers or scanners.

CPU : Pentium IV 2.4 GHz or above Memory (Primary) : 512 MB, 1 GB or above

Hard Disk : 40 GB, 80GB, 160GB or above

Monitor : 15 VGA color

**CHAPTER 3**

## SYSTEM MODELLING AND DESIGN

## 3.1 ER DIAGRAM

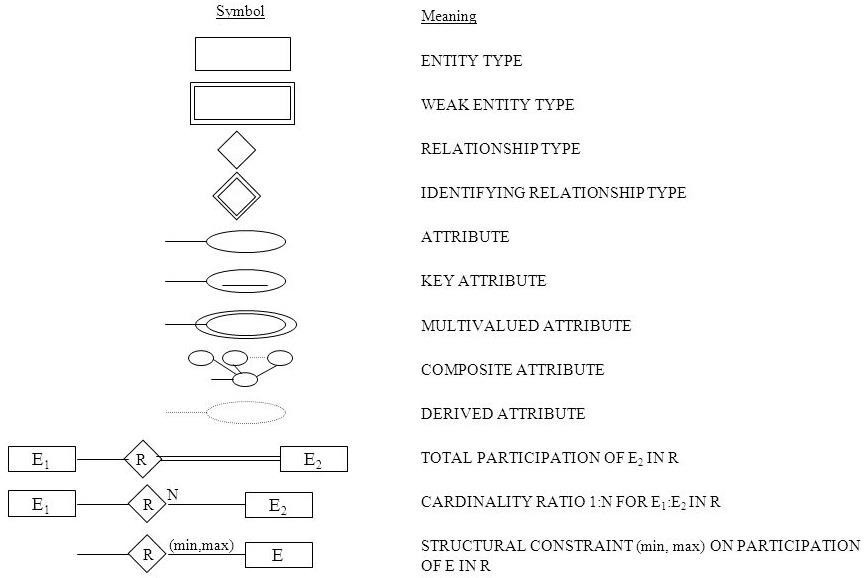
This chapter of the report describes the structure of the project, followed by Entity Relationship Diagram, Schema Diagram and the table structures.

***3.1.1* ER Diagram with relationships and cardinality ratio**

An entity relationship model*,* also called an entity-relationship (ER) diagram*,* is a graphical representation of entities and their relationships to each other, typically used in computing in regard to the organization of data within databases or information systems. An entity is a piece of data-an object or concept about which data is stored.

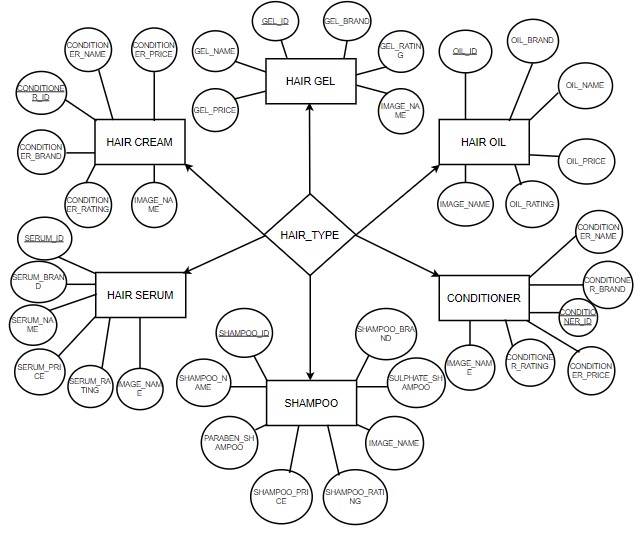
The cardinalit*y* or fundamental principle of one data aspect with respect to another is a critical feature. The relationship of one to the other must be precise and exact between each other in order to explain how each aspect links together. In simple words Cardinality is a way to define the relationship between two entities.

The following are the notations of the ER diagram:



**Fig 3.1: Notations for ER Diagrams**

The ER diagram below shows the relationship between the many tables that exist in the database for the functioning of **Hair Care Recommender.**

****

**Fig 3.2: ER Diagrams For Hair Care Recommender**

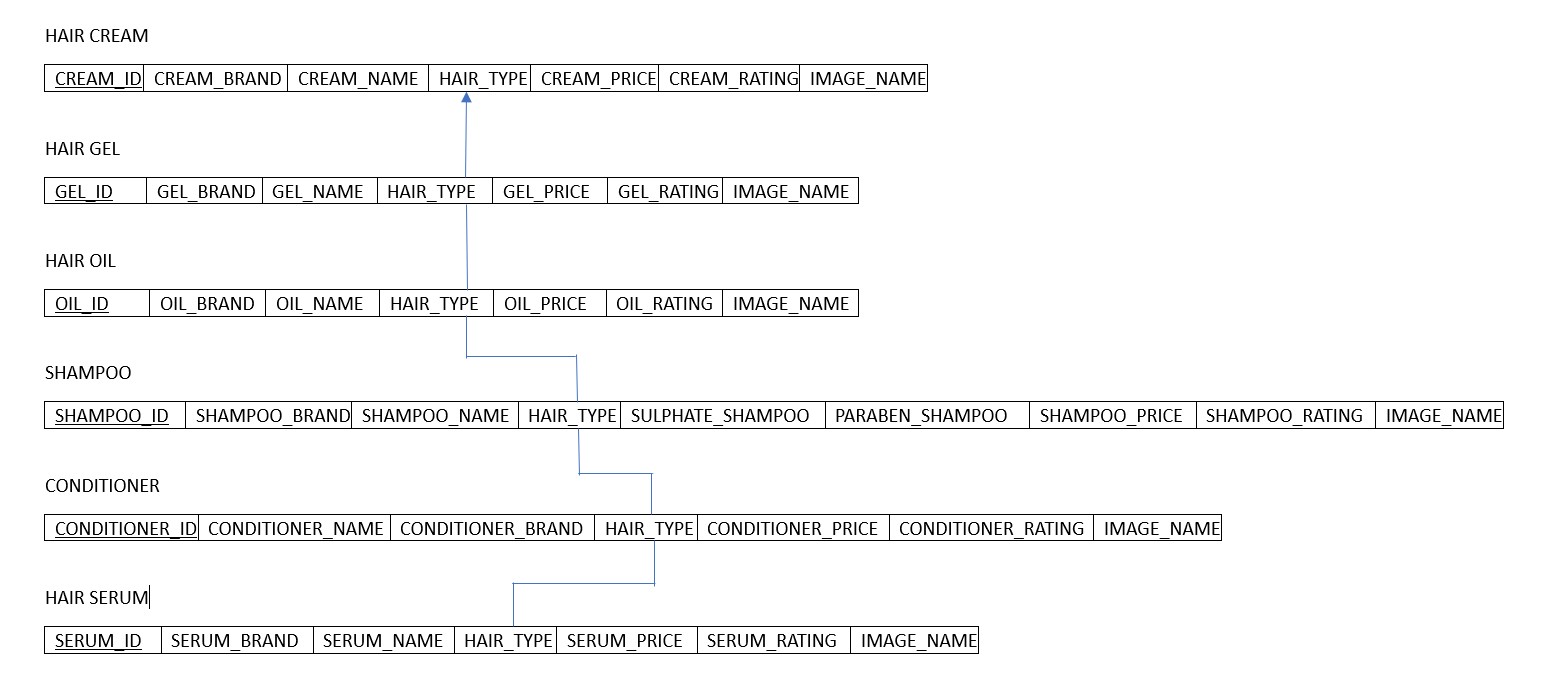
The Entity-Relationship (ER) diagram for hair products depicts a single entity: "Products." This entity encapsulates various attributes of hair care products, including ProductID (Primary Key), Name, Brand, Type, Ingredients, SuitableHairTypes, and SuitableConcerns. Each product is uniquely identified by a ProductID and characterized by its name, brand, type (e.g., shampoo, conditioner), ingredients, and suitability for different hair types and concerns. Unlike other ER diagrams, this one does not include any relationships with other entities, focusing solely on describing the attributes and properties of hair care products within the system.

This simplified representation allows for a concise overview of product-related information, facilitating efficient data management and retrieval for users seeking personalized hair care recommendations in the Hair Care Recommender project.

**3.2 SCHEMA DIAGRAM**

In any data model it is important to distinguish between the description of the database and the database itself. The description of a database is called the database schema, which is specified during database design and is not expected to change frequently.

A displayed schema is called a schema diagram. A schema diagram displays only some aspects of a schema, such as the names of record types and data items, and some types of constraints.



**Fig 3.2: Schema Diagram For Hair Care Recommender**

### In a schema diagram focusing solely on hair products, the diagram showcases the structure and

### attributes of the "Products" entity. Each product is represented as a rectangular box containing

### attributes such as Product ID, Name, Brand, Type, Ingredients, Suitable Hair Types, and

### Suitable Concerns. The Product ID serves as the primary key, uniquely identifying each product

### the system. Other attributes include the name of the product, its brand, type (e.g., shampoo,

### conditioner), ingredients, and suitability for different hair types and concerns. Unlike more

### complex schema diagrams, this simplified version doesn't display relationships with other entities. It provides a clear visual representation of the attributes associated with hair care products, aiding in efficient data organization and retrieval for personalized recommendations within the Hair Care Recommender project.

### 3.3 NORMALIZATION

Normalization, in the context of databases, is the process of organizing the attributes and tables of a relational database to minimize redundancy and dependency. It ensures that the database structure is efficient, avoids data anomalies, and maintains data integrity.

The most commonly discussed normal forms are:

1. **First Normal Form (1NF):** In 1NF, each column in a table must contain atomic (indivisible) values, and there should be no repeating groups of columns.
2. **Second Normal Form (2NF):** To achieve 2NF, a table must first satisfy 1NF, and then all non-key attributes (columns) must be fully functionally dependent on the entire primary key, not just part of it.
3. **Third Normal Form (3NF):** A table is in 3NF if it is in 2NF and all non-key attributes are dependent only on the primary key and not on other non-key attributes.

There are higher normal forms beyond 3NF, such as **Boyce-Codd Normal Form (BCNF), Fourth Normal Form (4NF), and Fifth Normal Form (5NF).** Each of these forms aims to eliminate different types of anomalies and redundancies in the database design.

Normalization helps ensure data integrity, reduces redundancy, and improves query performance in databases. The level of normalization applied depends on the specific requirements of the database and the trade-offs between storage efficiency and data integrity.

***3.3.1* Normalization Used In Our Database**

To determine if normalization has been used in your project, we would need to analyze the database schema and how data is stored and structured in your MySQL database (haircare in this case). Normalization is a database design technique used to organize data in a relational database efficiently.

Normalization typically involves dividing a database into two or more tables and defining relationships between them. The goal is to reduce redundancy and dependency in data, thereby improving data integrity and reducing the chances of anomalies.

Here are some signs that suggest normalization might have been applied in your project:

1. **Multiple tables:** If your database schema includes multiple tables, each containing different types of data (e.g., shampoo, conditioner, serum, oil, cream, gel), it's likely that normalization has been used to organize related data into separate entities.
2. **Elimination of redundant data:** Normalization involves reducing or eliminating redundant data by breaking down tables into smaller, related tables. If you have avoided storing the same information in multiple places, it indicates normalization.
3. **Use of primary and foreign keys:** In normalized databases, relationships between tables are established using primary and foreign keys. If your tables have primary keys (unique identifiers) and foreign keys that link related tables together, it suggests a normalized database design.
4. **Consistent data format:** Normalization often leads to consistent data formats within tables. Each table should store data relevant to a specific entity or concept, ensuring that the data is stored in a standardized format.
5. **Avoidance of update anomalies:** Normalization helps prevent update anomalies, such as insertion, deletion, and modification anomalies, by breaking down tables into smaller, related tables. If your database design minimizes the risk of such anomalies, it indicates normalization.

Review your database schema and how data is stored to determine if normalization principles have been applied. If you have divided your data into separate tables, established relationships between them, and minimized redundancy, then normalization has likely been used in your project.

## CHAPTER 4

## IMPLEMENTATION

**4.1 SOURCE CODE**

Source code is the human-readable version of a computer program written in a programming language. It consists of instructions and commands written by a programmer to tell a computer what to do. Source code needs to be translated into machine code, which computers can understand and execute, through a process called compilation or interpretation, depending on the programming language used. Source code is typically stored in text files with specific file extensions corresponding to the programming language, such as .java for Java, .py for Python, .cpp for C++, etc.

Below is the source code used for Hair Care Recommender

***4.1.1*Main Code**

from flask import Flask, render\_template, request, redirect, url\_for, session, jsonify

from flask\_mysqldb import MySQL

import random

from random import sample

app = Flask(\_\_name\_\_, static\_folder='static')

# MySQL Configuration

app.config['MYSQL\_HOST'] = 'localhost'

app.config['MYSQL\_USER'] = 'root'

app.config['MYSQL\_PASSWORD'] = 'beena123.'

app.config['MYSQL\_DB'] = 'haircare'

mysql = MySQL(app)

username = "admin"

password = "admin123"

# Define routes

@app.route('/')

def index():

return render\_template('homepage.html')

@app.route('/about')

def about():

return render\_template('about.html')

@app.route('/admin',methods=['GET', 'POST'])

def admin():

return render\_template('admin.html')

@app.route('/logout')

def logout():

session.pop('username', None)

return redirect(url\_for('/'))

@app.route('/login', methods=['GET', 'POST'])

def login():

if request.method == 'POST':

# Check username and password

# Add your authentication logic here

if request.form['username'] == username and request.form['password'] == password:

# Redirect to the admin page if login is successful

return redirect(url\_for('admin'))

else:

# Render the login page with an error message if login fails

return render\_template('login.html', error='Invalid username or password')

else:

# Render the login form

return render\_template('login.html')

@app.route('/add\_entry', methods=['POST'])

@app.route('/add\_entry', methods=['POST'])

@app.route('/add\_entry', methods=['POST'])

@app.route('/add\_entry', methods=['POST'])

def add\_entry():

if request.method == 'POST':

table\_name = request.form['table\_name']

# Retrieve data for new entry from the form

# Customize these lines to match your column names Insert data into the specified table

cur = mysql.connection.cursor()

if table\_name == 'shampoo':

s\_id=request.form['s\_id']

s\_brand=request.form['s\_brand']

s\_name = request.form['s\_name']

hair\_type=request.form['hair\_type']

sulphate\_shampoo = request.form['sulphate\_shampoo']

paraben\_shampoo = request.form['paraben\_shampoo']

s\_price = request.form['s\_price']

s\_rating = request.form['s\_rating']

image\_name = request.form['image\_name']

cur.execute("INSERT INTO shampoo (s\_id, s\_brand, s\_name, hair\_type,sulphate\_shampoo,paraben\_shampoo,s\_price, s\_rating,image\_name) VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s)", (s\_id, s\_brand, s\_name, hair\_type, sulphate\_shampoo, paraben\_shampoo, s\_price, s\_rating, image\_name))

elif table\_name == 'conditioner':

c\_id=request.form['c\_id']

c\_brand=request.form['c\_brand']

c\_name = request.form['c\_name']

hair\_type=request.form['hair\_type']

c\_price = request.form['c\_price']

c\_rating = request.form['c\_rating']

image\_name = request.form['image\_name']

cur.execute("INSERT INTO conditioner (c\_id, c\_brand, c\_name, hair\_type,c\_price, c\_rating,image\_name) VALUES (%s, %s, %s, %s, %s, %s, %s)", (c\_id, c\_brand, c\_name, hair\_type, c\_price, c\_rating, image\_name))

# Add more conditions for other tables if needed

elif table\_name == 'cream':

cr\_id=request.form['cr\_id']

cr\_brand=request.form['cr\_brand']

cr\_name = request.form['cr\_name']

hair\_type=request.form['hair\_type']

cr\_price = request.form['cr\_price']

cr\_rating = request.form['cr\_rating']

image\_name = request.form['image\_name']

cur.execute("INSERT INTO cream (cr\_id, cr\_brand, cr\_name, hair\_type,cr\_price, cr\_rating,image\_name) VALUES (%s, %s, %s, %s, %s, %s, %s)", (cr\_id, cr\_brand, cr\_name, hair\_type, cr\_price, cr\_rating, image\_name))

# Add more conditions for other tables if needed

elif table\_name == 'oil':

o\_id=request.form['o\_id']

o\_brand=request.form['o\_brand']

o\_name = request.form['o\_name']

hair\_type=request.form['hair\_type']

o\_price = request.form['o\_price']

o\_rating = request.form['o\_rating']

image\_name = request.form['image\_name']

cur.execute("INSERT INTO oil (o\_id, o\_brand, o\_name, hair\_type,o\_price, o\_rating,image\_name) VALUES (%s, %s, %s, %s, %s, %s, %s)", (o\_id, o\_brand, o\_name, hair\_type, o\_price, o\_rating, image\_name))

# Add more conditions for other tables if needed

elif table\_name == 'gel':

g\_id=request.form['g\_id']

g\_brand=request.form['g\_brand']

g\_name = request.form['g\_name']

hair\_type=request.form['hair\_type']

g\_price = request.form['g\_price']

g\_rating = request.form['g\_rating']

image\_name = request.form['image\_name']

cur.execute("INSERT INTO gel (g\_id, g\_brand, g\_name, hair\_type,g\_price, g\_rating,image\_name) VALUES (%s, %s, %s, %s, %s, %s, %s)", (g\_id, g\_brand, g\_name, hair\_type, g\_price, g\_rating, image\_name))

# Add more conditions for other tables if needed

elif table\_name == 'serum':

se\_id=request.form['se\_id']

se\_brand=request.form['se\_brand']

se\_name = request.form['se\_name']

hair\_type=request.form['hair\_type']

se\_price = request.form['se\_price']

se\_rating = request.form['se\_rating']

image\_name = request.form['image\_name']

cur.execute("INSERT INTO serum (se\_id, se\_brand, se\_name, hair\_type,se\_price, se\_rating,image\_name) VALUES (%s, %s, %s, %s, %s, %s, %s)", (se\_id, se\_brand, se\_name, hair\_type, se\_price, se\_rating, image\_name))

# Add more conditions for other tables if needed

mysql.connection.commit()

cur.close()

return redirect(url\_for('admin'))

@app.route('/questionnaire', methods=['GET', 'POST'])

def questionnaire():

if request.method == 'POST':

hair\_type = request.form['hair\_type']

return redirect(url\_for('index'))

return render\_template('index.html', question="What's your hair type?")

@app.route('/get-columns/<table\_name>', methods=['GET'])

def get\_columns(table\_name):

cur = mysql.connection.cursor()

cur.execute("SHOW COLUMNS FROM {}".format(table\_name))

columns = [row[0] for row in cur.fetchall()]

cur.close()

return jsonify(columns)

@app.route('/shampoo-info', methods=['POST'])

def get\_hair\_info():

hair\_type = request.form['hair\_type']

cur = mysql.connection.cursor()

cur.execute('SELECT s\_id, s\_brand, s\_name, hair\_type, sulphate\_shampoo, paraben\_shampoo, s\_price, s\_rating, image\_name FROM shampoo WHERE hair\_type = %s', (hair\_type,))

shampoo\_info\_list = cur.fetchall()

cur.execute('SELECT c\_id, c\_name, c\_brand,hair\_type, c\_price, c\_rating, image\_name FROM conditioner WHERE hair\_type = %s', (hair\_type,))

conditioner\_info\_list = cur.fetchall()

cur.execute('SELECT se\_id, se\_brand, se\_name, hair\_type, se\_price, se\_rating, image\_name FROM serum WHERE hair\_type = %s', (hair\_type,))

serums\_info\_list = cur.fetchall()

cur.execute('SELECT o\_id, o\_brand, o\_name, hair\_type, o\_price, o\_rating, image\_name FROM oil WHERE hair\_type = %s', (hair\_type,))

oil\_info\_list = cur.fetchall()

cur.execute('SELECT cr\_id, cr\_brand, cr\_name, hair\_type, cr\_price, cr\_rating, image\_name FROM cream WHERE hair\_type = %s', (hair\_type,))

cream\_info\_list = cur.fetchall()

cur.execute('SELECT g\_id, g\_brand, g\_name, hair\_type, g\_price, g\_rating, image\_name FROM gel WHERE hair\_type = %s', (hair\_type,))

gel\_info\_list = cur.fetchall()

cur.close()

if shampoo\_info\_list or conditioner\_info\_list or serums\_info\_list or oil\_info\_list or cream\_info\_list or gel\_info\_list:

random\_shampoos = sample(shampoo\_info\_list, min(2, len(shampoo\_info\_list)))

random\_conditioners = sample(conditioner\_info\_list, min(2, len(conditioner\_info\_list)))

random\_serums = sample(serums\_info\_list, min(2, len(serums\_info\_list)))

random\_cream = sample(cream\_info\_list, min(2, len(cream\_info\_list)))

random\_oil = sample(oil\_info\_list, min(2, len(oil\_info\_list)))

random\_gel = sample(gel\_info\_list, min(2, len(gel\_info\_list)))

return render\_template('shampoo\_info.html', hair\_type=hair\_type, shampoo\_info\_list=random\_shampoos, conditioner\_info\_list=random\_conditioners, serums\_info\_list=random\_serums, oil\_info\_list=random\_oil, cream\_info\_list=random\_cream, gel\_info\_list=random\_gel)

else:

return render\_template('shampoo\_info.html', hair\_type=hair\_type, shampoo\_info\_list=None, conditioner\_info\_list=None, serums\_info\_list=None, oil\_info\_list=None, cream\_info\_list=None, gel\_info\_list=None)

if \_\_name\_\_ == '\_\_main\_\_':

app.run(debug=True)

***4.1.2* SQL Code**

create database haircare;

CREATE TABLE shampoo (

s\_id INT PRIMARY KEY,

s\_brand VARCHAR(255),

s\_name VARCHAR(255),

hair\_type VARCHAR(255),

sulphate\_shampoo VARCHAR(20),

paraben\_shampoo VARCHAR(20),

s\_price VARCHAR(20),

s\_rating DECIMAL(3, 2),

image\_name VARCHAR(255)

);

CREATE INDEX idx\_hair\_type ON shampoo (hair\_type);

CREATE TABLE conditioner (

c\_id INT PRIMARY KEY,

c\_name VARCHAR(255),

c\_brand VARCHAR(255),

hair\_type VARCHAR(255),

c\_price VARCHAR(20),

c\_rating DECIMAL(3, 2),

image\_name VARCHAR(255),

FOREIGN KEY (hair\_type) REFERENCES shampoo(hair\_type)

);

CREATE TABLE cream (

cr\_id INT PRIMARY KEY,

cr\_name VARCHAR(255),

cr\_brand VARCHAR(255),

hair\_type VARCHAR(255),

cr\_price VARCHAR(20),

cr\_rating DECIMAL(3, 2),

image\_name VARCHAR(255),

FOREIGN KEY (hair\_type) REFERENCES conditioner(hair\_type)

);

CREATE TABLE oil (

o\_id INT PRIMARY KEY,

o\_brand VARCHAR(255),

o\_name VARCHAR(255),

hair\_type VARCHAR(255),

o\_price VARCHAR(20),

o\_rating DECIMAL(3, 2),

image\_name VARCHAR(255),

FOREIGN KEY (hair\_type) REFERENCES cream(hair\_type)

);

CREATE TABLE gel (

g\_id INT PRIMARY KEY,

g\_brand VARCHAR(255),

g\_name VARCHAR(255),

hair\_type VARCHAR(255),

g\_price VARCHAR(20),

g\_rating DECIMAL(3, 2),

image\_name VARCHAR(255),

FOREIGN KEY (hair\_type) REFERENCES oil(hair\_type)

);

CREATE TABLE serum (

se\_id INT PRIMARY KEY,

se\_brand VARCHAR(255),

se\_name VARCHAR(255),

hair\_type VARCHAR(255),

se\_price VARCHAR(20),

se\_rating DECIMAL(3, 2),

image\_name VARCHAR(255),

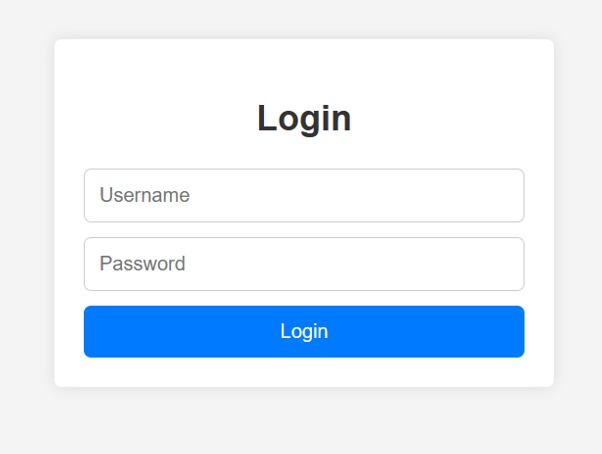
FOREIGN KEY (hair\_type) REFERENCES gel(hair\_type)

);

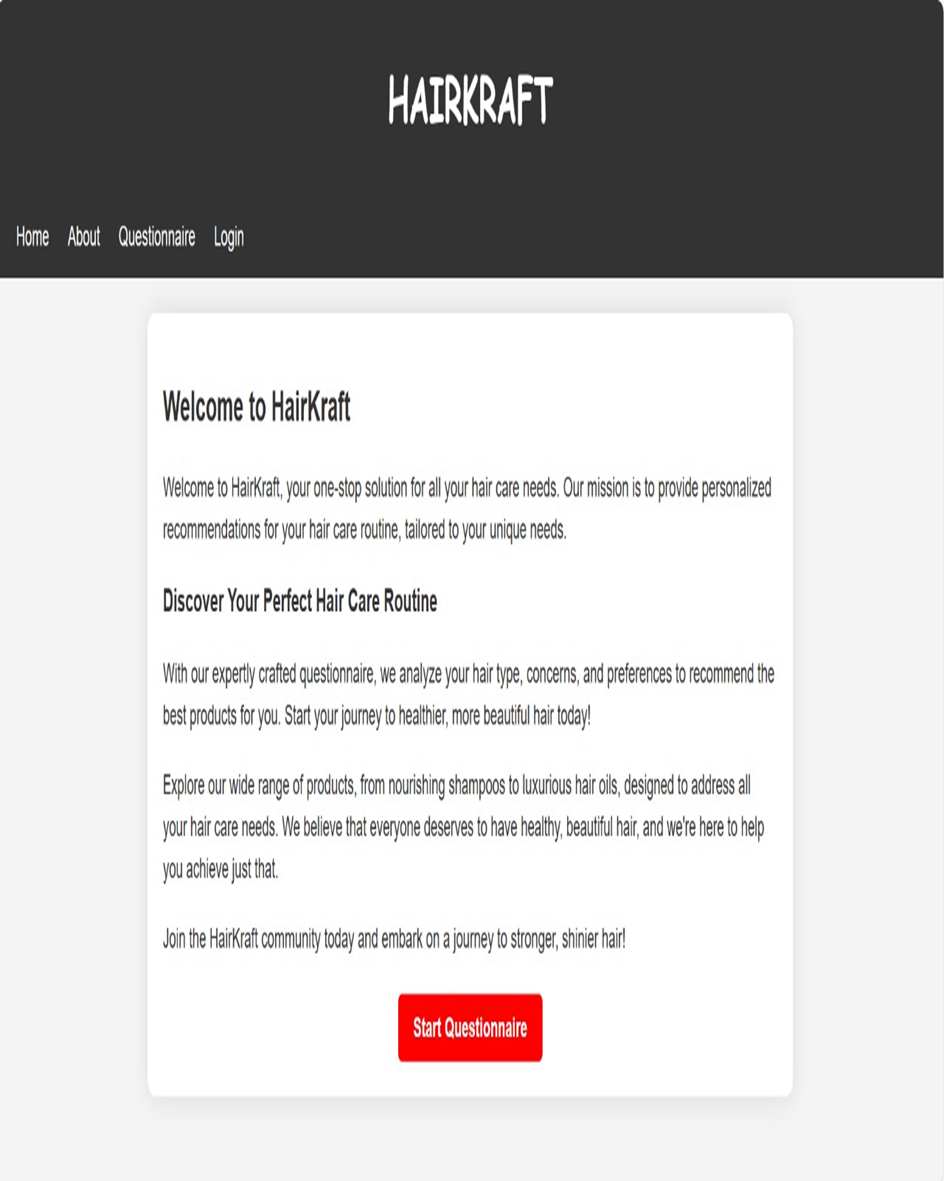
## CHAPTER 5

## OUTPUT

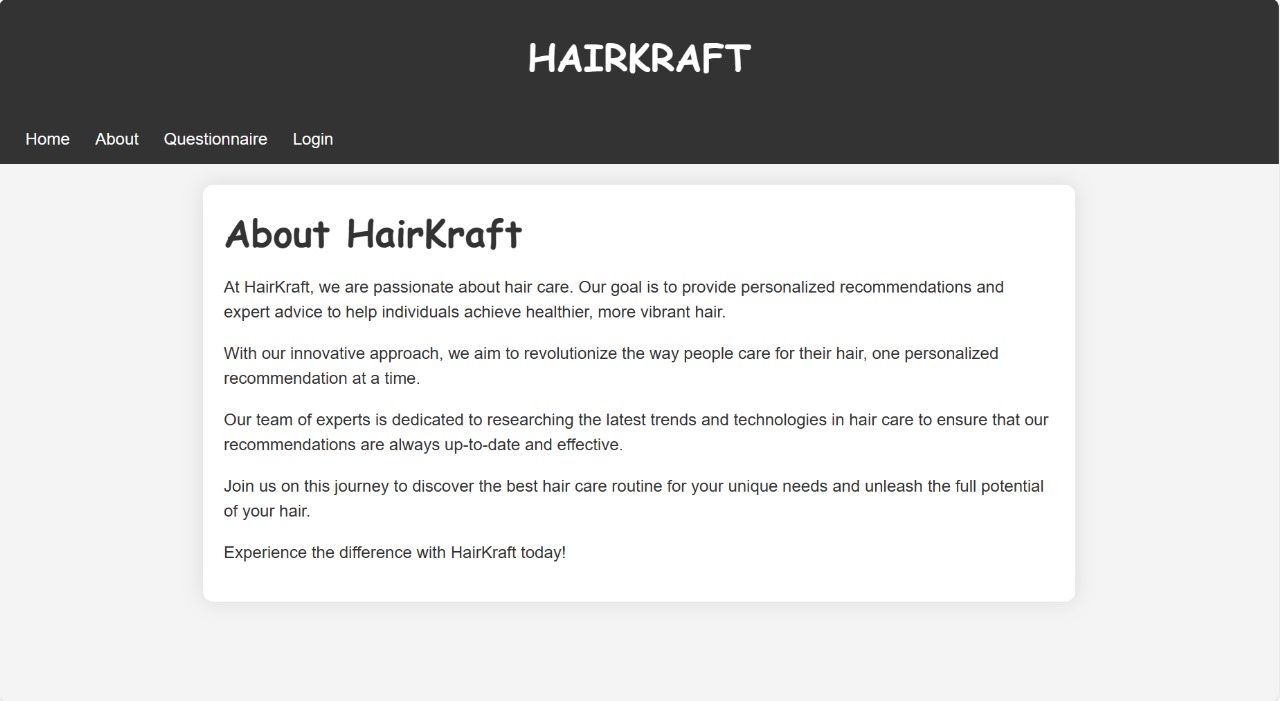
## 5.1 SCREENSHOTS

****

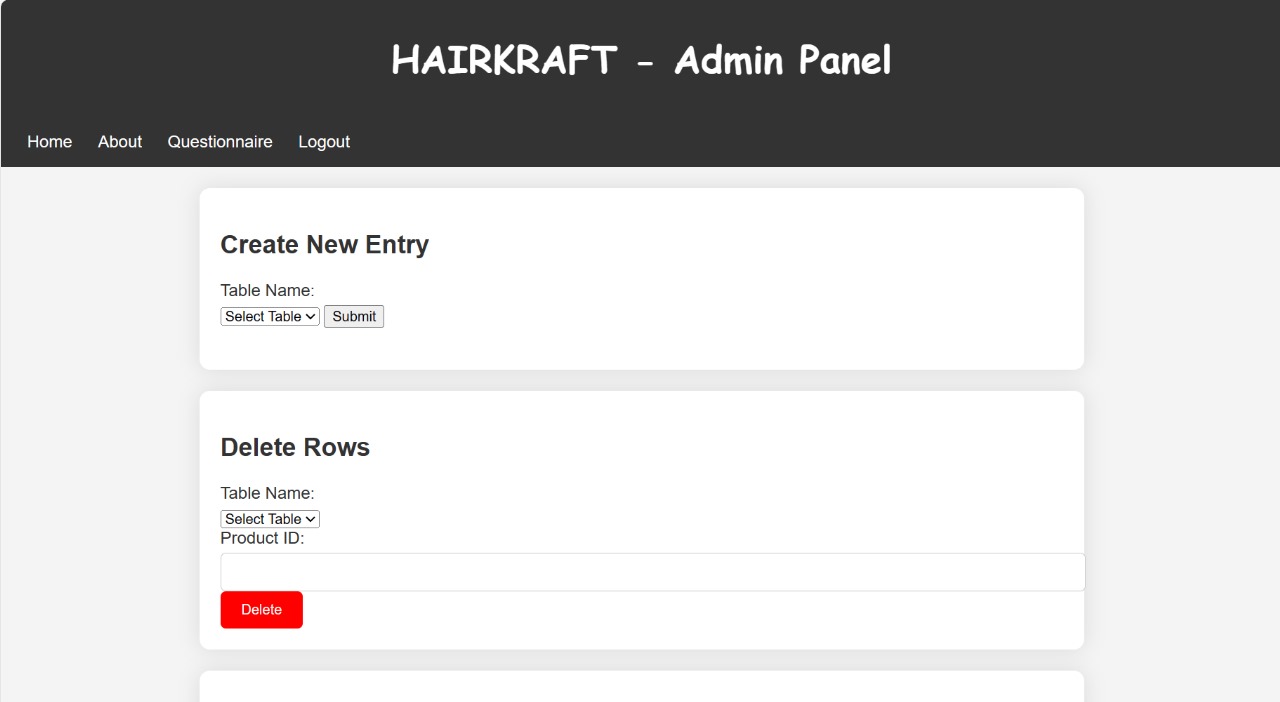
**Fig 5.1: Login Page**

****

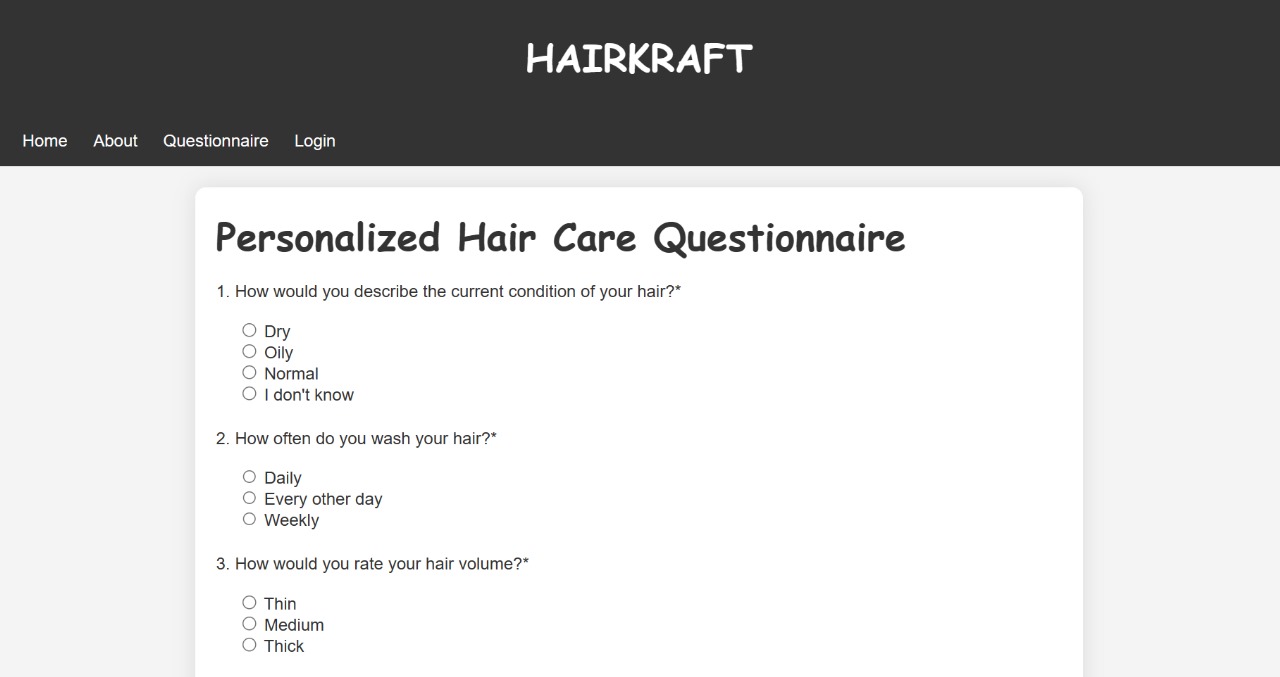
**Fig 5.2: Home Page**

****

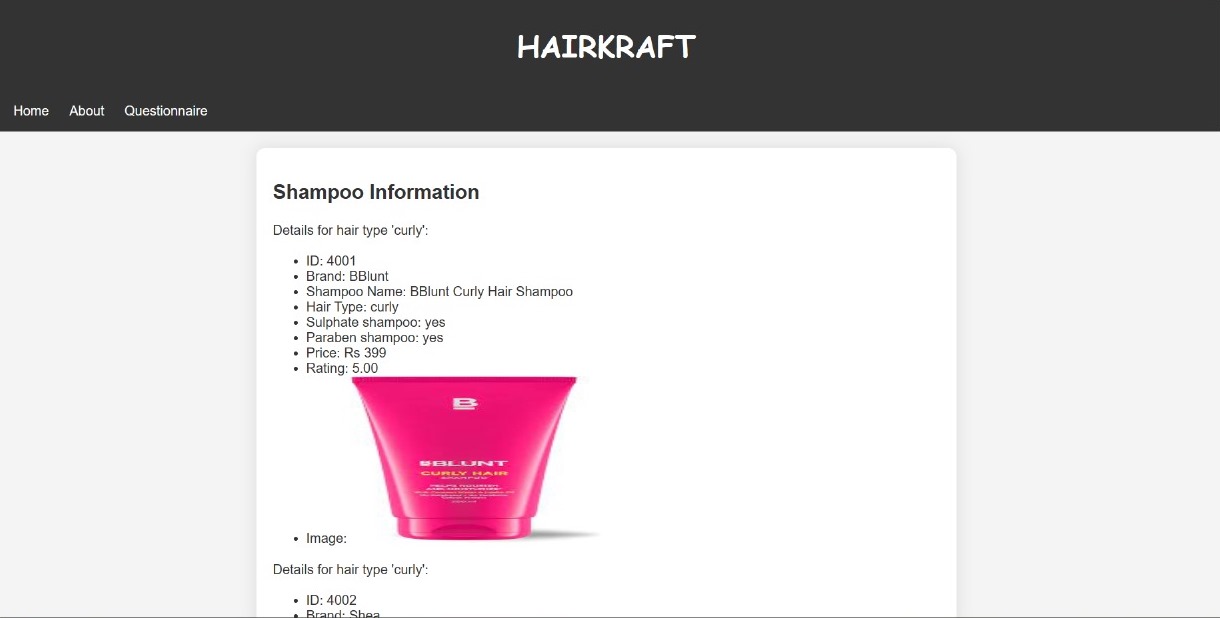
**Fig 5.3: About Page**

****

**Fig 5.4: Admin Page**

****

**Fig 5.5: Questionnaire Page**

****

**Fig 5.6: Result Page**

## CHAPTER 6

## CONCLUSION

In culmination, the Hair Care Recommender project signifies a significant stride towards personalized hair care solutions, underpinned by a robust database management system. By seamlessly amalgamating user-specific characteristics and preferences with a comprehensive product database, the system aspires to redefine the landscape of hair care recommendations. Through its intuitive user interface, users are empowered to make informed decisions, ensuring not only satisfaction but also fostering a sense of ownership over their hair care regimen.

The project's innovative approach, featuring sophisticated recommendation algorithms, holds promise in reshaping consumer behavior and industry standards. By providing tailored suggestions that resonate with individual needs and aspirations, the Hair Care Recommender project transcends the one-size-fits-all paradigm, paving the way for a more personalized and gratifying user experience.

Looking ahead, the project remains poised to evolve in tandem with technological advancements and changing market dynamics. Opportunities abound for further integration with e-commerce platforms, leveraging machine learning algorithms for dynamic updates, and expanding accessibility through mobile applications. Moreover, the potential for social media integration and the development of an analytics dashboard offer avenues for deeper engagement and continuous improvement.

In essence, the Hair Care Recommender project is not merely a technological endeavor but a catalyst for innovation and empowerment within the hair care industry. By prioritizing user-centricity and adaptability, it aspires to transcend the confines of traditional recommendation systems, championing a new era of personalized care and consumer satisfaction. As the journey continues, the project stands resolute in its commitment to excellence, poised to chart new frontiers and redefine the standards of excellence in hair care recommendation systems.