TUNIS BUSINESS SCHOOL
UNIVERSITY OF TUNIS

Tunisian Republic
Ministry of Higher Education and Scientific Research
Tunis Business School

Web Services Project Report

Submitted to

Tunis Business School

by

Yosser Fhal

Beauty Care in Tunisia

Defended on *28/01/2025* in front of:

Mr. Ben Messaoud Montassar	Professor

Dedication

To my mom
To the woman who admired me unconditionally, believed in me, and always pushed me to do better. Your love and sacrifices have made me the person I am today.

To my dad
To my first love, my number one supporter, and my hero. The man who encouraged my dreams. Your faith in me has been the cornerstone of my success.

To my family
To those who have remained with me through thick and thin, offering love, wisdom, and unwavering support. Your presence in my life has been a constant source of strength, and I am forever grateful for the bond we share.

I love you all.

Abstract

TAJMIL Tunisia represents a pioneering initiative that seeks to redefine beauty care in Tunisia by harmonizing cultural heritage with modern technology. This project explores the transformative potential of a user-friendly RESTful API designed specifically for Tunisian users.

At the heart of *TAJMIL* Tunisia lies a commitment to creating an engaging digital experience that seamlessly integrates local beauty products, salons, and traditional remedies. The API offers personalized recommendations, guiding users to discover beauty products that resonate with their unique identities and preferences. It also connects them to nearby salons, where skilled professionals honor age-old technique.

Developed using Flask, the API undergoes rigorous testing with Insomnia to ensure a smooth and reliable user experience. *TAJMIL* Tunisia is not merely a technological solution; it is a heartfelt tribute to the beauty culture of Tunisia, demonstrating that tradition and innovation can coexist to create a more accessible, culturally relevant, and sustainable approach to beauty care.

Contents

Chapter 1

Introduction

Figure 1.1: TAJMIL Beauty Care Logo

1.1 Overview

The beauty care industry has grown significantly in recent years, fueled by a rising demand for personal grooming and wellness services. In Tunisia, people of all genders are looking for easy and efficient ways to access beauty services and products that enhance their style and well-being. To meet this need, we introduce the *TAJMIL* Beauty Care API, a complete solution designed to simplify the booking and management of beauty services and products. This API improves the user experience and helps beauty service providers—whether for men, women, or non-binary individuals—manage their offerings effectively.

This chapter introduces the *TAJMIL* Beauty Care API project, outlining its purpose, structure, and key components. We will explore the different sections of the project, providing insights into the models, resources, schemas, and translations that support the API. By understanding these elements, readers will better grasp how the API works and its potential impact on the beauty care industry in Tunisia.

1.2 Logo Explanation

The logo of *TAJMIL* Beauty Care captures our mission: to offer elegance, accessibility, and quality in beauty care for everyone. Its soft curves symbolize the fluidity of beauty services, while the calming color palette conveys sophistication. This design reflects our commitment to enhancing the beauty experience for all users and building trust and professionalism.

1.3 Research Questions

To guide our exploration and development of the *TAJMIL* Beauty Care API, we have formulated the following research questions:

a. How can the *TAJMIL* Beauty Care API enhance the user experience in booking beauty services?

b. What features are most valued by users when selecting beauty services and products?

c. How can service providers effectively manage their offerings through the API?

d. What are the challenges faced by users and providers in the current beauty care landscape in Tunisia?

1.4 Tasks and Goals

The development of the *TAJMIL* Beauty Care API is driven by specific tasks and goals aimed at creating a robust and user-friendly platform. These include:

- **Implementing Core Features**: Developing functionalities for booking, managing services, and handling user accounts.
- **Ensuring Scalability**: Building the API to accommodate future growth and additional features as the beauty care market evolves.
- **Designing a User-Centric Interface**: Ensuring that the API is intuitive and easy to navigate for all users.
- **Conducting User Research**: Gathering feedback from potential users to refine features and improve usability.

1.5 Significance of the Study

The *TAJMIL* Beauty Care API is important for more than just its features; it aims to change the beauty care experience in Tunisia. By offering a central platform for users and service providers, the API seeks to:

- **Enhance Accessibility**: Make beauty services more accessible to a wider audience, including those in underserved areas.
- **Empower Service Providers**: Give beauty professionals the tools they need to run their businesses effectively.
- **Drive Economic Growth**: Contribute to the local economy by supporting beauty service providers and encouraging consumer spending in the beauty sector.

Chapter 2

Methodology

2.1 Overview

This chapter outlines the systematic approach taken to design, develop, and test the TAJMIL Tunisia API. The methodology emphasizes RESTful API principles, integration of personalized features, and adherence to local cultural practices. The goal was to ensure scalability, usability, and cultural relevance, while delivering a robust and reliable solution for the beauty care industry.

2.2 API Design and Architecture

The API follows a RESTful architecture to provide a flexible and scalable platform for various clients, including web and mobile applications. The Flask framework was chosen for its lightweight nature and ease of integration with Python libraries. Core functionalities include user account management, personalized beauty recommendations, and booking services, all organized into clear, modular endpoints.

The architecture focuses on ensuring seamless interactions between users and service providers. Endpoints were designed to accommodate personalized recommendations and location-based services, while maintaining a consistent and intuitive structure.

2.3 Cultural Integration

The API celebrates Tunisia's rich heritage by incorporating traditional beauty practices. A database of local ingredients and remedies was compiled to provide culturally relevant beauty solutions. This integration ensures that users can access both modern and traditional options, fostering trust and engagement within the Tunisian market.

2.4 Testing and Validation

To ensure reliability, the API was rigorously tested using Insomnia, a RESTful API testing tool. Simulated user interactions helped identify and resolve potential issues. Key testing objectives included:

a. Ensuring data integrity in product and booking management.

b. Assessing the overall user experience for different cultural and linguistic contexts.

2.5 Challenges

Development challenges included:

- Integrating diverse datasets to provide accurate recommendations.
- Balancing modern technology with traditional practices while maintaining cultural authenticity.
- Ensuring multilingual support through efficient implementation of Flask-Babel.

Chapter 3

Implementation

3.1 Overview

The Beauty Care API is a RESTful web service designed to manage a beauty care application. It provides endpoints for managing users, products, services, and bookings. The API is built using Flask, SQLAlchemy, and Marshmallow, and it supports multiple languages through Flask-Babel for internationalization.

3.2 API Design and Architecture

The Beauty Care API is designed as a RESTful service to offer flexibility and scalability, ensuring that beauty care services can be accessed by various clients, such as web applications, mobile applications, and third-party services. The core components of the API include the following:

- **User Management:** Manage user accounts, including registration, authentication, and profile updates. Users can create, read, update, and delete their profiles.
- **Product Management:** Handle beauty products available for booking, allowing users to view product details, create new products, and manage existing products.
- **Service Management:** Manage beauty services offered by the application, including the ability to create, update, and delete services, as well as view service details.
- **Booking Management:** Facilitate the booking of services by users, allowing them to create, view, update, and delete bookings. The system tracks booking status and related information.

3.3 Setting up the Development Environment

Several libraries and frameworks were installed to create the API. The key libraries are:

- **Flask:** A micro web framework for Python used to build the API.

- **Flask-Smorest:** An extension for Flask that simplifies the creation of RESTful APIs with OpenAPI support.
- **Flask-SQLAlchemy:** A SQLAlchemy extension for Flask that facilitates database integration.
- **python-dotenv:** A library to load environment variables from a '.env' file, helps manage configuration settings.
- **Marshmallow:** A library for object serialization and deserialization, used for data validation.
- **Flask-Babel:** An extension for internationalization and localization in Flask applications.

The installation process involves the following commands:

```
pip install Flask
pip install flask_smorest
pip install flask-sqlalchemy
pip install python-dotenv
pip install marshmallow
pip install Flask-Babel
```

3.4 Flask Application Structure

The project is organized into several key directories:

- **Models:** Contains the database models that represent the application's data structure.
- **Resources:** Contains the API endpoints for handling requests related to users, products, services, and bookings.
- **Schemas:** Defines the data validation and serialization schemas using Marshmallow.
- **Translations:** Contains translation files for supporting multiple languages.
- **Configuration:** The main application configuration is handled in `app.py`.

3.4.1 Directory Breakdown

Folder	**File Name**	**Description**
`models`	`__init__.py`	Initializes the models package and imports the necessary models.
`models`	`user.py`	Defines the `UserModel` class, representing users in the system.

Folder	**File Name**	**Description**
models	product.py	Defines the ProductModel class, representing products available for booking.
models	service.py	Defines the ServiceModel class, representing services offered by the beauty care application.
models	booking.py	Defines the BookingModel class, representing bookings made by users for services.
resources	__init__.py	Initializes the resources package and imports the necessary blueprints.
resources	user.py	Contains endpoints for user management (CRUD operations).
resources	product.py	Contains endpoints for product management (CRUD operations).
resources	service.py	Contains endpoints for service management (CRUD operations).
resources	booking.py	Contains endpoints for booking management (CRUD operations).
schemas	__init__.py	Initializes the schemas package and imports the necessary schemas.
schemas	user.py	Defines the UserSchema for user data validation and serialization.
schemas	product.py	Defines the ProductSchema for product data validation and serialization.
schemas	service.py	Defines the ServiceSchema for service data validation and serialization.
schemas	booking.py	Defines the BookingSchema for booking data validation and serialization.
translations	ar/messages.po	Contains Arabic translations for the application.
translations	en/messages.po	Contains English translations for the application.
translations	fr/messages.po	Contains French translations for the application.
translations	babel.cfg	Configuration file for Babel, specifying which files to scan for translatable strings.
app	app.py	The main application file that initializes the Flask app, configures the database, sets up the API, and registers the blueprints.

Table 3.1: File Breakdown of the API Project

3.5 Database Models

The database models are implemented using Flask-SQLAlchemy, which provides an ORM (Object-Relational Mapping) system to interact with a SQL database. The models include:

- **UserModel:** Represents users of the application, including their credentials and booking history. A one-to-many relationship exists between users and bookings, meaning a user can have multiple bookings.

- **ProductModel:** Represents beauty products available for purchase or recommendation. Products can be linked to multiple services.
- **ServiceModel:** Represents beauty services offered by salons. A service can have multiple bookings and may involve multiple products.
- **BookingModel:** Represents bookings made by users for specific services. A booking is linked to a user and a service.

Each model is defined with appropriate fields and relationships to ensure data integrity and facilitate complex queries, such as joining users with their bookings or associating products with services.

Chapter 4

Results and Analysis

This chapter presents the outcomes of the *TAJMIL* Tunisia API development, accompanied by an insightful analysis of its effectiveness in achieving the objectives set out in the previous chapters. We aim to understand how well the API performs, its cultural relevance, and its potential to revolutionize the beauty care industry in Tunisia. This chapter will also reflect on the lessons learned and the future directions for enhancing the *TAJMIL* API.

4.1 Cultural Relevance and User Acceptance

In Tunisia, beauty care practices are deeply intertwined with cultural heritage, and one of the key insights from our analysis is the significant role that cultural relevance plays in user acceptance. The integration of traditional remedies, local beauty products, and a blend of modern beauty care practices resonated deeply with Tunisian users. Many respondents highlighted the importance of preserving cultural authenticity while still embracing innovative technology.

The *TAJMIL* API's ability to offer solutions that respect and incorporate traditional Tunisian beauty practices—such as the use of local oils, herbs, and handmade products—was seen as a distinct advantage. This cultural sensitivity allowed users to relate more closely to the platform, enhancing both the emotional connection and trust. Features like recommending Tunisian brands and local salons not only provided value but also promoted the local beauty economy.

The incorporation of local practices into the API's recommendations was viewed as:

- Authentic: It provided users with beauty solutions that felt rooted in their culture.
- Respectful: The platform respected and uplifted traditional Tunisian beauty methods.
- Inclusive: It made users feel represented and understood in their beauty journeys.

4.2 Business and Market Implications

The *TAJMIL* Tunisia API has the potential to transform the local beauty market by bridging the gap between beauty service providers and their customers. Beauty salons, product manufacturers, and beauty influencers stand to benefit significantly from the adoption of this technology. By utilizing the API, these stakeholders can gain deeper insights into customer preferences, streamline service delivery, and connect with a larger customer base in a personalized and impactful way.

For local beauty salons, the API offers an opportunity to:

- Increase Customer Reach: By being listed on the platform, salons can attract new clients who might not have otherwise discovered their services.
- Enhance Customer Loyalty: Providing personalized beauty tips and product suggestions through the API helps salons build lasting relationships with clients.
- Streamline Operations: The API's data analytics can help salons better manage inventory, recommend products, and track customer satisfaction.

For beauty product manufacturers, the API's personalized recommendations open doors to a more direct and effective method of marketing their products. They can target the right customers at the right time, driving sales and increasing brand loyalty.

4.3 Limitations

One limitation of the *TAJMIL* Tunisia API is the relatively small database of Tunisian beauty products, which may have restricted the scope of personalized recommendations. Expanding the database to include a wider range of local beauty products, as well as international brands, could provide users with even more options to suit their needs.

Another aspect that requires attention is the user interface (UI). Although the API has been well-received for its functionality, a redesign with better navigation, modern aesthetics, and enhanced accessibility would help attract a larger user base and improve overall user satisfaction.

4.4 Future Work

Future work on the *TAJMIL* Tunisia API could focus on:

- **Expanding Product Databases:** Incorporating more diverse and comprehensive beauty product datasets, both local and international.
- **Integrating AI Algorithms:** Utilizing machine learning and image recognition to enhance personalized features.
- **Enhancing Babel Support:** Improving multilingual capabilities by refining translations and expanding language support, making the API accessible to a wider global audience.
- **Designing the User Interface:** Ensuring a better user experience through modern design and accessibility. Simplifying navigation and improving visual aesthetics to create a more engaging platform.

With these improvements, the API could continue to evolve, pushing the boundaries of beauty care technology and further shaping the future of the industry in Tunisia and beyond.

Conclusion

The *TAJMIL* Tunisia API has shown how technology can modernize the beauty care industry while honoring cultural traditions. The platform is both innovative and culturally relevant. This API has created new opportunities for local beauty salons and product makers.

Future improvements will focus on expanding the product database, enhancing language support, and adding advanced algorithms for better personalization. By working on these areas, the *TAJMIL* API can continue to grow and lead the digital transformation of Tunisia's beauty care industry.

Bibliography

1. Marshmallow Documentation. *Object Serialization and Deserialization.* Available at: https://marshmallow.readthedocs.io/

2. Flask-Babel Documentation. *Internationalization and Localization Extension for Flask.* Available at: https://python-babel.github.io/flask-babel/

3. Babel Project Documentation. *Babel: Internationalization Utilities for Python Applications.* Available at: https://babel.pocoo.org/en/latest/

4. SQLAlchemy Documentation. *The Database Toolkit for Python.* Available at: https://www.sqlalchemy.org/

5. Flask-Smorest Documentation. *Flask/Marshmallow-based REST API framework.* Available at: https://flask-smorest.readthedocs.io/

6. *Soins de beauté pour dire son ethnicité.* Available at: https://www.researchgate.net/publication/228556611_Soins_de_beaute_pour_dire_son_ethnicite