YAŞAR UNIVERSITY
FACULTY OF ENGINEERING
DEPARTMENT OF COMPUTER ENGINEERING

COMP4910 Senior Design Project 1, Fall 2023
Advisor: Assoc. Prof. Dr. Ömer ÇETİN

GLOWG: Personalized Skin Care Powered by AI
Final Report

15.01.2025

By:
Ceren Sude Yetim, 21070001045
İrem Demir, 20070001029
Gizem Tanış, 20070001047
Ece Topuz, 21070001057

PLAGIARISM STATEMENT

This report was written by the group members and in our own words, except for quotations from published and unpublished sources which are clearly indicated and acknowledged as such. We are conscious that the incorporation of material from other works or a paraphrase of such material without acknowledgement will be treated as plagiarism according to the University Regulations. The source of any picture, graph, map or other illustration is also indicated, as is the source, published or unpublished, of any material not resulting from our own experimentation, observation or specimen collecting.

Project Group Members:

Name, Lastname	Student Number	Signature	Date
Ceren Sude Yetim	21070001045		20.01.2025
İrem Demir	20070001029		20.01.2025
Ece Topuz	21070001057		20.01.2025
Gizem Tanış	20070001047		20.01.2025

Project Advisors:

Name, Lastname	Department	Signature	Date
Ömer Çetin	Computer Engineering		20.01.2025

ACKNOWLEDGEMENTS

We would like to express our sincere gratitude to all those who supported and contributed to the success of the *Glow Genie* project.

Firstly, we extend our deepest thanks to our project advisor, Assoc. Prof. Dr. Ömer ÇETİN, whose guidance, expertise, and valuable feedback played a crucial role in shaping the direction of this project. Their support was instrumental in overcoming challenges.

We would also like to acknowledge the collaborative efforts of our team members, İrem Demir, Ceren Sude Yetim, Ece Topuz, Gizem Tanış, whose dedication, hard work, and creative ideas were key to the shaping the direction of this project. Each member's commitment and contribution were essential in achieving the project goals.

KEYWORDS

Skin Care
Product Suitability Evaluation
Product Recommendation System
Machine Learning
GPT Integration
Dataset Preprocessing
Model Training
Ingredient-Based Classification
Data Analysis
Feature Engineering
Web-Based Application
User Interface Design
System Testing and Validation

ABSTRACT

The GlowGenie project, developed as part of the Senior Design Project 1 (COMP 4910) at Yaşar University, seeks to transform the skincare routine creation process by introducing a sophisticated web-based recommendation system. GlowGenie provides personalized skin care product suggestions tailored to users' unique skin types, tones, and preferences, addressing the challenge of selecting suitable products from an overwhelming market. By integrating advanced machine learning algorithms and GPT-based ingredient analysis, the project ensures that users receive scientifically informed and user-specific recommendations.

During the Fall 2023 semester, the project team concentrated on the foundational planning, research, and design phases. This included extensive research on skin types and product compatibility, designing a questionnaire to predict skin types, and creating comprehensive Requirements Specification and Design Specification Documents. Additionally, the team developed UML diagrams, database structures, and functional prototypes, focusing on a scalable, secure, and user-friendly system architecture.

GlowGenie introduces a novel approach to skincare by integrating a real-time database update system and a machine learning model to evaluate and classify products based on ingredient suitability. By providing insightful feedback on why a product suits or does not suit a user’s skin type, GlowGenie aims to educate users while improving their skincare journey. This project highlights the potential of AI-driven technology to enhance personalized healthcare experiences and simplify complex decision-making processes in the beauty and wellness industry.

ÖZET

GlowGenie projesi, Yaşar Üniversitesi'nde Senior Design Project 1 (COMP 4910) kapsamında geliştirilmiş olup, kişiye özel cilt bakım rutini oluşturma sürecini dönüştürmeyi amaçlayan, gelişmiş bir web tabanlı öneri sistemidir. GlowGenie, kullanıcıların cilt tipi, cilt tonu ve tercihlerine göre özel olarak uyarlanmış cilt bakım ürünü önerileri sunarak, pazardaki ürün çeşitliliği arasında uygun ürün seçimi sorununu çözmeyi hedefler. Proje, makine öğrenimi algoritmalarını ve GPT tabanlı içerik analizini entegre ederek, bilimsel verilere dayalı ve kullanıcı odaklı öneriler sunar.

2023 Güz dönemi boyunca proje ekibi, planlama, araştırma ve tasarım aşamalarına odaklanmıştır. Bu süreçte cilt tipleri ve ürün uygunluğu üzerine kapsamlı araştırmalar yapılmış, cilt tipini tahmin etmeye yönelik bir anket tasarlanmış ve kapsamlı Gereksinim Şartnamesi ile Tasarım Şartnamesi belgeleri hazırlanmıştır. Ayrıca, ekip UML diyagramları, veri tabanı yapıları ve işlevsel prototipler geliştirerek ölçeklenebilir, güvenli ve kullanıcı dostu bir sistem mimarisine odaklanmıştır.

GlowGenie, ürünlerin içerik uygunluğuna göre değerlendirilip sınıflandırılması için gerçek zamanlı bir veri tabanı güncelleme sistemi ve makine öğrenimi modeli entegre ederek yenilikçi bir yaklaşım sunmaktadır. Bir ürünün neden kullanıcı cilt tipi için uygun veya uygun olmadığını açıklayan geri bildirimler sağlayarak, kullanıcıları bilgilendirmeyi ve cilt bakım sürecini geliştirmeyi amaçlamaktadır. GlowGenie projesi, AI destekli teknolojinin kişiselleştirilmiş sağlık ve güzellik deneyimlerini geliştirme ve karmaşık karar alma süreçlerini basitleştirme potansiyelini gözler önüne sermektedir.

TABLE OF CONTENTS

LIST OF ACRONYMS/ABBREVIATIONS

UML: Unified Modeling Language
ML: Machine Learning
API: Application Programming Interface
REST: Representational State Transfer
SQL: Structured Query Language

1. INTRODUCTION

1.1. Description of the Problem

In today's beauty industry, selecting skincare products tailored to individual needs has become increasingly complex. With a vast array of options available in the market, consumers often face difficulties in identifying products suitable for their unique skin types, tones, and preferences. This challenge is compounded by the lack of reliable, personalized systems that guide users in making informed decisions. Traditional approaches, such as generic product reviews and broad recommendations, fail to address the specific requirements of individual users and often lead to trial-and-error purchases, wasting time and resources.

Moreover, the ingredients in skincare products are frequently updated by manufacturers, making it challenging for users to stay informed about the compatibility and safety of products. Existing recommendation platforms do not dynamically adapt to changes in product formulations, nor do they provide detailed explanations for why a product is suitable or unsuitable. Additionally, users seeking allergen-free or targeted skincare solutions often struggle to find products that align with their preferences.

This gap in the market presents an opportunity to develop a robust system that not only recommends products based on personal attributes but also keeps product information up to date and provides transparent feedback about ingredient suitability.

1.2. Project Goal(s)

The GlowGenie project seeks to address these challenges by developing an intelligent, web-based skincare recommendation platform. This system will utilize machine learning models to classify products based on their compatibility with different skin types, skin tones, and user-specified allergens. It aims to provide users with personalized product recommendations that are visually intuitive, highlighting suitable and unsuitable options.

Key goals of the project include:

- Developing a machine learning-based model to evaluate and label skincare products according to their suitability for various skin types.
- Ensuring ingredient data remains accurate by periodically updating product information via GPT integration.
- Offering detailed feedback on why a product is suitable or unsuitable, including the ingredients responsible for the classification.
- Creating a user-friendly interface that allows users to filter products by categories and view personalized recommendations seamlessly.
- Enabling users to update their profiles, including skin type, skin tone, and allergens, to refine recommendations further.

By transforming the skincare selection process into a personalized, efficient, and transparent experience, GlowGenie aims to empower users with the tools and knowledge to make better skin care choices, ultimately enhancing their skincare journey.

1.3. Project Outputs/Deliverables

Project deliverables for COMP 4910 are listed below:
i. Project Assignment Form (PAF)
ii. RSD: Requirements Specifications Document, v1.0 (textual form)
iii. Final Report
iv. RSD: Requirements Specifications Document, v2.0 (in UML), also as Appendix A of Final Report
v. DSD: Design Specifications Document, v1.0 (High Level Design in UML), also as Appendix B of Final Report
vi. Project Poster (The first draft)
vii. Project Presentation (The first draft)
viii. Project Web (The first draft, must contain at least the poster and presentation)
ix. Peer Evaluation Form, one form per team member

Predicted project deliverables for COMP 4920 are listed below:
i. Project Assignment Form
ii. Final version of Requirements Specifications Document iii. Final version of Design Specifications Document
iv. Final version of Project Poster
v. Final version of Project Presentation
vi. Final version of Project Web
vii. Project Manual
viii. Final Report, together with appendices ix. Project Source Code

2. SURVEY OF RELATED WORK

2.1 Academic Papers

Several academic papers have contributed to the development of personalized skincare systems by applying data mining techniques, machine learning algorithms, and dermatological expertise to classify skin types and recommend skincare products. Below are some key academic papers relevant to the field.

1. **"A Personalized Skincare Recommendation System Using Machine Learning"**
   Authors: Supriya Jadhav, Disha Memane, Kadambaree Supekar, Shraddha Shinde, Trupti Jadhav

   DOI:10.52783/cienceng.v11i1.127
   Summary: This paper explores the application of machine learning algorithms for building a personalized skincare recommendation system. By analyzing user data such as skin type, concerns, and product preferences, the system provides tailored skincare product suggestions.
   Key Contribution: Demonstrated the potential of machine learning to personalize skincare recommendations based on user profiles.

2. **Intelligent Facial Skin Care Recommendation System**

   Authors: B. Lokesh, Anjali Devarakonda, G Srinivas, Nitish Kumar Naik

DOI: 10.33472/AFJBS.6.Si2.2024.1822-1830

Summary: This paper introduces an **Intelligent Facial Skin Care Recommendation WebApp** that uses Convolutional Neural Networks (CNN) to classify skin types and offer personalized skincare advice. The system analyzes user-uploaded skin images, considering attributes like dryness, oiliness, and sensitivity, to recommend tailored products. It aims to provide more accurate and individualized skincare solutions compared to traditional systems.

Key Contribution: Introduced the concept of integrating both dermatological expertise and user preferences for building effective skincare product recommendations.

2.2 Commercial Solutions/Products

Source: Proactiv+

Description: Proactiv+ is a well-known commercial skincare brand that offers personalized acne treatment regimens. Their online questionnaire helps users identify their skin concerns, and they recommend a specific set of products based on individual needs.

Key Contribution: Proactiv+ uses personalized recommendations for treating acne, relying on both product knowledge and individual skin assessments.

Source: Curology

Description: Curology offers customized skincare treatments based on skin type and issues, including acne, aging, and other skin concerns. Users submit photos of their skin, which are reviewed by licensed dermatologists who prescribe a personalized skincare formula.

Key Contribution: Provides personalized formulations that address specific skin issues, such as acne and aging. Offers telemedicine consultations for a more personalized experience.

3. REQUIREMENTS

The project began with the creation of a Project Assignment Form (PAF), which included details like a project code, title, team information, and a summary of the project or product. This was followed by the development of the first version of the Requirements Specification Document (RSD 1.0), which was written in a structured text format. Subsequently, an updated version, RSD 2.0, was generated. This version enhanced the initial requirements outlined in RSD 1.0 and presented them using the Unified Modeling Language (UML) notation.

The definitive requirements for our COMP4910 project are detailed in Appendix A, under the title "Requirements Specifications Document, version 2.0.

4. DESIGN

4.1. High Level Design

The high-level design of the GlowGenie system utilizes a **Layered Architecture** to ensure modularity, maintainability, and scalability. This architecture separates the system into three key layers: the **Presentation Layer**, which handles user interactions through a dynamic and responsive React.js interface; the **Application Layer**, which processes business logic, integrates with machine learning models, and facilitates API communication; and the **Data Layer**, which manages persistent storage using PostgreSQL for efficient and structured data handling.

This architecture was selected because it promotes a clear separation of concerns, enabling ease of future development, adaptability for new features, and enhanced maintainability of the codebase. Detailed specifications for the high-level design can be found in **Appendix B: Design Specifications Document**, specifically in sections B.3.1 (**GLOWG Software System Architecture**).

4.2. Detailed Design

This section will actually be completed in COMP 4920.

4.3. Realistic Restrictions and Conditions in the Design

In the design of the **Glow Genie** project, several restrictions and conditions were considered to ensure the system is functional and feasible within the project's scope. These include the following:

Data Security: The current design does not include advanced security measures like data encryption or multi-factor authentication, as the system's primary focus is on providing personalized skincare recommendations. However, security measures such as user authentication and basic data privacy will be included in the future phases of development.

Security and Authentication: While basic security measures are implemented, there are limitations in advanced security protocols. For instance, the system employs standard password enforcement rather than multifactor authentication, a decision driven by the need to balance security with user convenience.

User Experience: As the system is designed to be user-friendly, it incorporates simple input methods and provides real-time feedback. However, it does not offer a complex user interface with multiple language support, as this is outside the current scope.
System Performance: The system is designed to handle a reasonable load, with a focus on ensuring quick response times for user input. Performance optimizations such as caching is not included at this stage but will be considered during later implementation phases.

5. IMPLEMENTATION AND TESTS

5.1. Implementation of the Product

This section will actually be written and completed in COMP 4920.

For the realization of the **Glow Genie** project in COMP 4920, a few techniques, tools, and technologies:

1. **Backend Development**: The backend will likely use **Python**, **Flask** due to their ease of integration with AI models like **ChatGPT** and their efficiency in handling user input and requests.
2. **AI Integration**: The primary AI model for personalized skincare recommendations will be **ChatGPT**. We'll leverage **OpenAI's API** to integrate the model, allowing the system to process user input and generate tailored suggestions.
3. **Frontend Development**: The frontend will be developed using **React** or **Angular**, depending on which framework best aligns with the team's expertise and project needs. **Bootstrap** might be used for UI components to ensure a clean and responsive design.
4. **Database**: For storing user data and skincare recommendations, a **relational database** like **PostgreSQL** is likely to be used for structured data management, ensuring easy retrieval and updating of records.

These implementation considerations are initial thoughts and will be further refined and detailed in the next phase of the project, COMP 4920. The choices of technologies and tools are based on their ability to meet the project requirements and objectives effectively, as initially outlined in the Design Specifications Document.

5.2. Tests and Results of Tests

These implementation considerations are initial thoughts and will be further refined and detailed in the next phase of the project, COMP 4920.

Unit tests for core functionalities. Integration tests for seamless interaction between components.Real-world testing to assess user experience and scalability.

6. PROJECT MANAGEMENT

6.1. Project Plan

The project plan for Glow Genie during the Fall semester has been carefully structured to ensure the successful completion of the project's initial phase. This plan outlines a series of key activities, each with defined start and end weeks, as well as a specified duration, ensuring that the project progresses smoothly. The detailed Project Plan is provided in Appendix C1 of the report. Below is a summary of the key activities and their schedules:

Initial Phase (Fall 2024):

1. **PAF Production**:
   The Project Approval Form (PAF) officially marked the commencement of the Glow

Genie project. This activity spanned the first two weeks of the semester, setting the foundation for project objectives and deliverables.

2. **Survey of Related Work**:
   This task involved conducting extensive research into existing personalized skincare systems, AI-driven recommendations, and skincare technologies. It was scheduled over three weeks (week 6, 7 and 10), forming a crucial part of the project's initial research phase.
3. **RSD 1.0 Production**:
   The first versions of the Requirements Specification Document (RSD) was developed over fourteen weeks. This document outlined the functional and non-functional requirements of the Glow Genie system, including user interaction, AI integration, and recommendation engine features.
4. **RSD 2.0 Revision**:
   A subsequent tree-week period was allocated to revise and refine the RSD, incorporating feedback, new insights, and the latest research findings in personalized skincare systems.
5. **DSD 1.0 Production**:
   The initial Design Specification Document (DSD) was produced over four weeks, outlining the proposed system architecture, including AI models, user interface design, and system components.
6. **Final Report Production**:
   Spanning the last two weeks of the semester, this task involved compiling and synthesizing all project activities, research findings, and documentation into the final report, ready for submission.
7. **Project Management Activities**:
   Throughout the semester, regular team meetings, progress tracking, risk management, and resource allocation were carried out to ensure timely delivery and smooth project progress.

Next Phase (Spring 2025):

1. **RSD v2.0 Revision**:
   The first task of the Spring semester will be to revise the RSD based on insights gained from the initial semester's work, focusing on refining the functional and non-functional requirements for system implementation.
2. **DSD v1.0 Revision**:
   The next phase will involve revising the DSD to ensure alignment with the updated requirements and incorporating any new technological advancements.
3. **DSD v2.0 as Detailed Design**:
   The detailed design phase will involve finalizing the DSD and preparing it for the implementation phase. This will include finalizing the architecture and AI models.
4. **Implementation and Testing Activities**:
   This critical phase will focus on the development of the Glow Genie system, followed by rigorous testing of its functionalities to ensure the system is reliable, accurate, and provides satisfactory recommendations for users.
5. **Project Management (PM)**:
   Ongoing project management activities will include monitoring, controlling, and closing the project phases to ensure successful delivery and integration of all components.

6.2. Project Effort/Manpower

This section is presented in Appendices C2 and C3.

6.3. Project Cost Analysis

Currently, there have been no hardware or software purchases for the **Glow Genie** project. All necessary tools and resources are being utilized through free and open-source platforms, as well as available academic licenses. Should any purchases be required in the future, they will be assessed and documented accordingly.

7. CONCLUSIONS

7.1. Summary

The **Glow Genie** project, initiated as part of the Senior Design Project, focuses on providing personalized skincare recommendations using AI, particularly **ChatGPT**, to help individuals choose suitable skincare products for their skin types.

Here is a summary and discussion of what has been accomplished so far:

1. **Planning**: The project started with identifying the need for a personalized skincare system. The team outlined the project's scope, objectives, and deliverables, followed by detailed planning.
2. **Requirements**: A Requirements Specification Document (RSD) was developed, outlining the functional and non-functional requirements of the system. This document serves as the foundation for the entire project and ensures alignment with identified needs.
3. **Research and Analysis**: The team conducted thorough research on existing skincare solutions and studied the integration of **ChatGPT** for personalized recommendations. Additionally, a detailed review of various **machine learning algorithms** was performed to enhance the recommendation system.
4. **System Design and Architecture**: The team worked on designing the system's architecture, focusing on creating a scalable and secure platform.
5. **Frontend and Backend Planning**: Preliminary discussions on the frontend and backend structure were conducted. The project team outlined the initial plan for implementing the user interface and integrating AI-driven functionality into the backend.
6. **Design Phase**: The Design Specification Document (DSD) was produced, detailing the proposed system architecture, design considerations, and technology stack. This phase focused on creating a scalable and secure design that would support the complex functionalities of the system.
7. **Documentation**: The project documentation, including the RSD, DSD initial designs, and planning drafts, has been compiled. This ensures a solid foundation for further development and provides a well-documented process for future stages.

To summarize, the **Glow Genie** project has made significant progress in the initial phase. The research, planning, and system design have laid the groundwork for a personalized skincare

system. The next steps involve implementing the system's core features, refining the AI integration, and preparing for testing and deployment.

7.2. Benefits of the Project

Benefits for Users:

- **Personalized Skincare:** Glow Genie provides tailored skincare product recommendations based on individual skin types, ensuring users select products that are safe and effective for their skin.
- **Informed Decision-Making:** The system helps users avoid harmful products and reduces the risk of skin irritation, enhancing overall skincare routines.
- **Cost Savings:** By suggesting only suitable products, it prevents unnecessary purchases, saving users money and time spent on trial and error.

Benefits for the Environment:

- **Reduced Product Waste:** By helping users choose products that match their skin types, Glow Genie minimizes product returns and unused products, contributing to less waste.
- **Sustainable Consumption:** Encourages mindful consumption by reducing the likelihood of users purchasing products that do not fit their needs, promoting sustainability in the skincare industry.

7.3. Future Work

To be Completed in COMP 4920:

Detailed Design: Finalize the Design Specification Document (DSD v2.0) by refining the system architecture, database schema, backend structure, and UI design. Implement core functionalities including user input processing, recommendation engine integration, product database setup, and user profile management to handle personalized skincare suggestions.

Machine Learning Algorithms: Enhance the recommendation engine by incorporating more advanced machine learning models. Optimize these models to provide more personalized and accurate recommendations, while enabling continuous learning from user feedback and data.

User Interface and Experience Enhancements: Refine the user interface based on user feedback to ensure it is intuitive, visually appealing, and mobile-responsive. Improve user experience (UX) through interactive elements and ensure ease of navigation and clarity in product recommendations.

Possible Additions Post-COMP 4920:

Support for Multiple Languages: Localizing the system to support multiple languages, catering to a global user base.

REFERENCES

1. Jadhav, S., Memane, D., Supekar, K., Shinde, S., & Jadhav, T. (2021). *A Personalized Skincare Recommendation System Using Machine Learning*. *CIENCIENGN*, 11(1). DOI: 10.52783/cienceng.v11i1.127
2. Lokesh, B., Devarakonda, A., Srinivas, G., & Naik, N. K. (2024). *Intelligent Facial Skin Care Recommendation System*. *AFJBS*, 6(Si2), 1822-1830. DOI: 10.33472/AFJBS.6.Si2.2024.1822-1830
3. Saidah, S., Fuadah, Y.N., Alia, F., Ibrahim, N., Magdalena, R., Rizal, S.: Facial skin type classification based on microscopic images using convolutional neural network (CNN). In: Proceedings of the 1st International Conference on Electronics, Biomedical Engineering, and Health Informatics: ICEBEHI 2020, 8–9 Oct, Surabaya, Indonesia, pp. 75–83. Springer Singapore (2021) DOI: 10.1007/978-981-33-6926-9_7
4. Kumar, K., Sinha, V., Sharma, A., Monicashree, M., Vandana, M.L., Vijay Krishna, B.S.: AIassisted college recommendation system. In: Intelligent Sustainable Systems: Proceedings of ICISS 2022, pp. 141–150. Springer Nature Singapore, Singapore (2022) DOI:10.1007/978-981-99-9018-4_28
5. Vinutha, M., Dayananda, R.B., Kamath, A.: Personalized skincare product recommendation system using content-based machine learning. In: 2024 4th International Conference on Intelligent Technologies (CONIT), pp. 1–9. IEEE, Bangalore, India (2024). DOI: 10.1109/CONIT61985.2024.10626458

APPENDICES

APPENDIX A: REQUIREMENTS SPECIFICATION DOCUMENT

COMP4910 Senior Design Project 1, Fall 2024
Advisor: Assoc. Prof. Dr. Ömer ÇETİN

GLOWG: Personalized Skin Care Powered by AI
Requirements Specifications Document

12.01.2025
Revision 2.0

By:
Ceren Sude Yetim, 21070001045
İrem Demir, 20070001029
Gizem Tanış, 20070001047
Ece Topuz, 21070001057

Revision History

Revision	Date	Explanation
1.0	10.11.2024	Initial requirements
1.1	09.12.2024	Made some adjustments on 2.1, 2.2, 2.3, 2.6, 2.7.
1.2	24.12.2024	● GlowG introduction title edited. ● 2.2 The registration form takes email, password and password verification information. Added email uniqueness check. ● 2.6 User preferences made clear, made additions to the user interface “For each step besides serum user can choose 1, for serum step user can choose from 1 to 3 products among 5 recommended products.“ ● Added UML diagrams to functions 2.2, 2.3, 2.4, 2.5, 2.6 and 2.7.
1.3	30.12.2024	2.3 Test questions updated. 3.1 Algorithms changed, instead of decision tree decided on random forest, and added clustering algorithms.
2.0	12.01.2025	Since the project’s purpose has changed, revised the whole document accordingly.

Table of Contents

1. Introduction to GlowGenie

Problem Definition:

Skin care is an area of great importance for both the physical health and aesthetic concerns of individuals. However, correctly identifying skin type and choosing products that are suitable for this information can be a complex and confusing process for most individuals. Difficulties in determining skin type often lead to the use of the wrong products, which can have negative effects on the skin such as irritation, dryness, oiliness, and acne. Wrong product choices not only endanger the health of the skin, but also lead to financial losses and users losing confidence in skin care.

The fast pace of modern life causes individuals to not have enough time to access accurate information, while the variety of products on the market makes it even more difficult for users to make choices. Moreover, users who do not have sufficient information about the content of many products and the effects of these contents on skin types are often forced to make choices based on advertisements and guidance. This can lead users to make unconscious decisions and use products that will negatively affect skin health.

Problem Solution:

With the increasing demand for skin care, GlowGenie is designed as an artificial intelligence-supported skin care platform that aims to facilitate individuals' access to accurate and reliable solutions. This platform has an artificial intelligence model that allows users to accurately determine their skin type. Although skin type information is of critical importance in choosing the right products, many people do not know their skin type or evaluate it incorrectly. To solve this problem, GlowGenie offers an intelligent system that analyzes users' skin types with simple questions. Thus, users can make a conscious and safe start to their skin care journey.

Another basic function of GlowGenie is to offer a model that evaluates users' current or potential product content in terms of suitability for their skin type. This model analyzes the content of the products and determines whether they are compatible with their skin type. At the same time, when the user requests a recommended product from the application, the application can also respond to this request. Products in the desired categories are recommended to the user in accordance with the skin type.

At this point, the need for technological tools that offer reliable, fast and personalized solutions is increasing. GlowGenie, The developed artificial intelligence-supported system facilitates the processes of determining skin type, evaluating product compatibility according to the person's skin type and providing recommendations for personal needs. GlowGenie offers a solution that users can choose with confidence, making skin care a more accessible, more effective and more user-friendly experience. By providing users with a reliable source of information and recommendations, it enables them to make more conscious and effective choices in their skin care decisions.

Literature Review:

AI-based recommendation systems have gained popularity due to their ability to provide personalized suggestions with high accuracy. For example, Kumar et al. [1] developed a college recommendation system using a content-based approach that matches user profiles with college profiles. Similarly, in skincare, machine learning models, like the CNN developed by Saidah et al. [2], are used to classify skin types and recommend products accordingly.

The increasing demand for personalized skincare recommendations has led to the development of content-based systems. Vinutha et al. [3] proposed a system that not only considers the chemical composition of products but also adjusts based on users' skin types and preferences. This system, like other AI models, focuses on user-specific features to provide more tailored suggestions, similar to how GlowGenie uses machine learning for skincare recommendations.

Jadhav et al. [4] also explore the potential of machine learning for building a personalized skincare recommendation system. Their system analyzes user data such as skin type, concerns, and product preferences to provide tailored skincare product suggestions. This aligns with the trend of leveraging user data to create highly personalized experiences in skincare.

Additionally, the Intelligent Facial Skin Care Recommendation WebApp proposed by Lokesh et al. [5] uses Convolutional Neural Networks (CNN) to classify skin types based on uploaded images. The system then recommends tailored skincare products based on attributes such as dryness, oiliness, and sensitivity. This concept of integrating both dermatological expertise and user preferences for building effective skincare product recommendations further enhances the accuracy and personalization of the system.

These developments suggest that content-based filtering methods are increasingly being applied across industries, including skincare, to create personalized experiences. As seen with the systems by Jadhav et al. [4] and Lokesh et al. [5], such approaches are likely to influence recommendation systems in other fields as well.

Challenges Addressed by GlowGenie:

The GlowGenie app overcomes several common challenges in creating effective skincare that vary from person to person.

With countless skincare products on the market containing a variety of active ingredients, users often struggle to determine which products and ingredients are best suited to their skin type. Factors such as skin type, sensitivities, allergies, and specific skincare goals add to the complexity of skincare. GlowGenie aims to overcome these challenges by making users an active part of the process, facilitating the process by encouraging users to share basic information about their skin type (oily, dry, combination, normal), known allergies, and product preferences. This information is processed by the app's ML models, allowing for skin-type-specific analyses for each user, creating a solid foundation for personalized skincare.

Another major challenge is the difficulty users face in assessing the compatibility of product ingredients with their skin type. Based on ingredient analysis, GlowGenie identifies products that are compatible with their skin type and makes recommendations that minimize possible side effects. The app also offers customized recommendations based on users' preferences in specific product categories (cleansers, moisturizers, sunscreens, toners, serums).

Personalized Approach:

GlowGenie offers a personalized approach that shapes skin care processes according to the needs of individuals. Each individual's skin structure and skin care goals may differ. For this reason, GlowGenie aims to be a platform that deeply understands users' needs and offers them special solutions. GlowGenie's personalized approach is based on machine learning models developed to learn users' skin type, allergies, and product preferences. As a first step, a machine learning model is activated that helps users correctly determine their skin type. This model analyzes users' skin type correctly and enables them to choose products that suit their needs.

In the subsequent stage, GlowGenie assesses the ingredients of the products that users are currently using or considering. By analyzing the composition of these products, the system provides detailed information regarding their ingredients and offers feedback on the suitability of the products for the user's specific skin type. This evaluation ensures that users make informed decisions about their skincare choices.

GlowGenie's personalized recommendation system also supports users in creating customized skin care routines according to their needs. For example, if the user is looking for a cleanser, GlowGenie only recommends cleansers suitable for their skin type. If there is a specific allergy, these factors are also taken into consideration and possible side effects are minimized.

Conclusion:

Although skin care is an important area for the health and aesthetic appearance of individuals, choosing the right product is a complex and time-consuming process. GlowGenie is an artificial intelligence-powered platform that offers solutions to these challenges, allowing users to accurately analyze their skin type and choose the most suitable products for their skin. With its personalized approach and smart recommendation system, GlowGenie aims to provide an effective and accessible skin care experience that users can safely choose.

2. Functional Requirements

2.1. Main User Interface and Functions

Main User Interface (Main Menu) and Functions are as follows:

User Account Operations

I want to log in or register for an account.
I want to update my profile information (skin type, skin color, allergens,etc.).
I want to take or retake the skin type test.
I want to log out.

Product Operations

I want to check if specific products are suitable for my skin type.
I want to view product details, including ingredients and get feedback on the product’s suitability.
I want to generate products that are suitable for my skin's characteristics, with the product categories I have chosen.
I want to view all the products and filter them by their categories.

2.2. Enter Registration Information

Users must register to use the app. Registration includes providing personal information and optional preferences.

User Registration Form:
The user will fill out a form containing the following information. Fields marked with a '*' are mandatory.

- Email*
- Password*
- Confirm Password*
- Skin Type (select from oily, dry, normal, combination, or "I don't know")*
- Skin Color (Light skin, Medium Skin, Dark Skin)*
- Allergens (optional text box)
- Submit & Cancel Buttons

Submit:

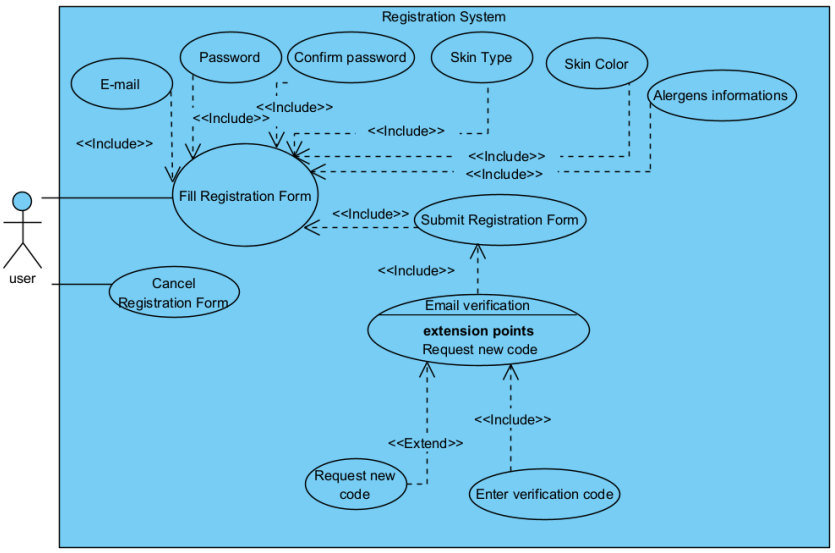
- Check if the email is unique.
- If unique, enter into the email verification process.
- If the email is already registered, it displays an error message: "*This email address is already registered.*"
- Checks if all the mandatory areas are filled, if not displays an error message: "*Please fill the mandatory areas.*"
- Checks if allergen input is in the desired form if not displays an error message: "*Please enter the allergens in the desired form. Such as Paraben, Alcohol, etc…*"
- Checks if the password includes at least one uppercase letter, one lowercase letter, one number, and one special character; otherwise, displays an error message: "*Password must contain at least one uppercase letter, one lowercase letter, one number, and one special character.*"

Cancel: Clears all fields and redirects to the main page.

Email Verification Process:

- The system sends a 4-digit confirmation code to the user's email. Displays a message "*We have sent you an email so we can verify your account. Could you please check your email address?"* The user must enter the code to proceed, and if the entered code does not match the sent code, an error message is displayed: *"Confirmation code is incorrect."* Users also have the option to request a new code if necessary. However, to prevent misuse: If a user requests a new code more than twice within 10 minutes, the system displays the message: "*Please wait 10 minutes before requesting a new code.*" If the user enters the wrong code three times, the system displays the message: "*You have entered the wrong code three times. Please request a new code.*" Once the correct code is submitted, a countdown of 7 seconds starts and the system displays a message "*Your email address has been verified, you can log in."* redirects the user to the home page.

Use Case Diagram



The provided use case diagram offers a detailed overview of the user registration process, highlighting essential actions, dependencies, and optional extensions. It effectively illustrates the interactions between the user and the registration system, focusing on core registration activities, validation processes, and optional actions.

Key Actors and Components:

- **User:** The primary actor who interacts with the system by filling out and submitting the registration form or canceling the process.
- **System Use Cases:**
    - **Fill Registration Form:** The user initiates the registration by providing necessary information, such as a unique email, password, confirmation password, skin type, skin color, and allergen information. Each of these fields is included in the form through <> relationships.
    - **Submit Registration Form:** After completing the form, the user submits it. This action triggers the system to validate the provided data, ensuring all required fields are filled, the email is unique, and allergens are correctly specified.
    - **Email Verification:** Upon submission, the system sends a verification code to the user's email. The user must enter this code to verify their email address. This process is essential for account activation.
    - **Enter Verification Code:** This mandatory step requires the user to input the received verification code. The system validates the code to complete the registration.
    - **Request New Code (Extension Point):** If the user does not receive the verification code or loses it, they can request a new code. This optional action is represented as an <> relationship in the diagram, providing flexibility during the email verification process.
    - **Cancel Registration Form:** The user can cancel the registration process at any stage, which redirects them back to the main page, ensuring user control over the process.

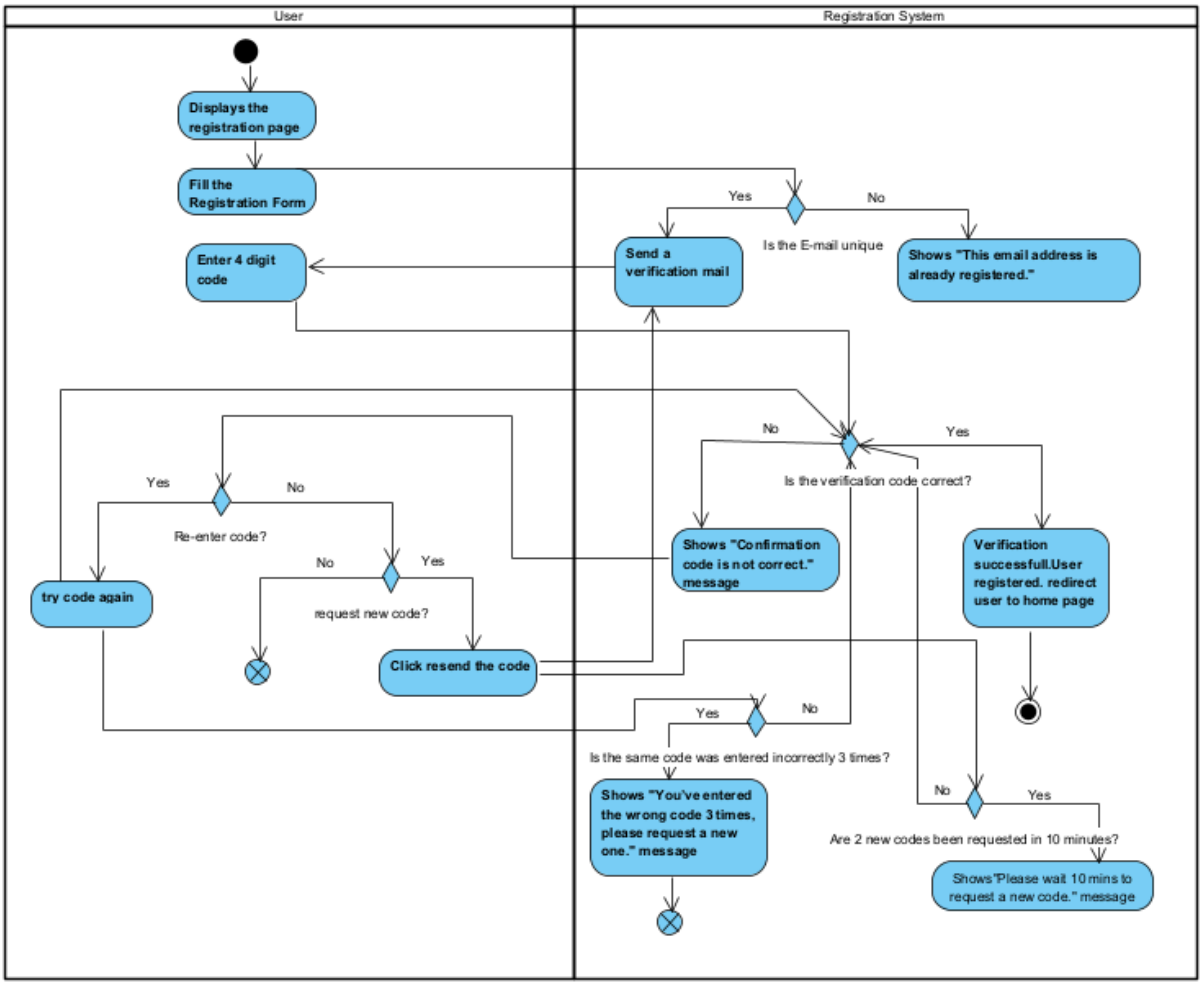
Use Case Diagram Relationships:

- **Include Relationships:** These represent mandatory steps within the registration process, such as entering an email, password, and other necessary information when filling out the form, as well as validating the verification code during the email verification step.
- **Extend Relationships:** These capture optional or alternative flows, such as the ability to request a new verification code if the original one is not received.

Summary:

This use case diagram provides a comprehensive view of the registration system, covering form completion, submission, validation, and email verification. The inclusion of optional extension points, like requesting a new verification code and canceling the registration, enhances the user experience by offering flexibility and control. The diagram effectively demonstrates system behavior and user interaction, ensuring a structured and user-friendly registration process.

Activity Diagram:



The diagram illustrates the user registration process, divided into two key sections: User and Registration System. The User section represents actions and decisions made by the user, while the Registration System section outlines the processes and responses managed by the system.

The process begins with the Initial Display and Registration Form. When the registration page is displayed, the user fills out the registration form and submits it. At this point, the system checks whether the provided email address is unique. If the email is unique, the system sends a verification email containing a 4-digit code to the user. If the email is already registered, the system displays an error message: “This email address is already registered.”

Next is the Verification Code Entry stage. The user enters the 4-digit verification code received in the email. The system then checks if the entered code matches the one sent. If the code is correct, the registration is completed successfully, and the user is redirected to the home page. If the code is incorrect, the system displays an error message: “Confirmation code is not correct.”

In cases of incorrect code entry, the process moves to Handling Incorrect Code Attempts. If the code is incorrect, the user has two options: they can re-enter the code or request a new one by clicking the “resend the code” button. This allows users flexibility in resolving issues without interrupting the registration process.

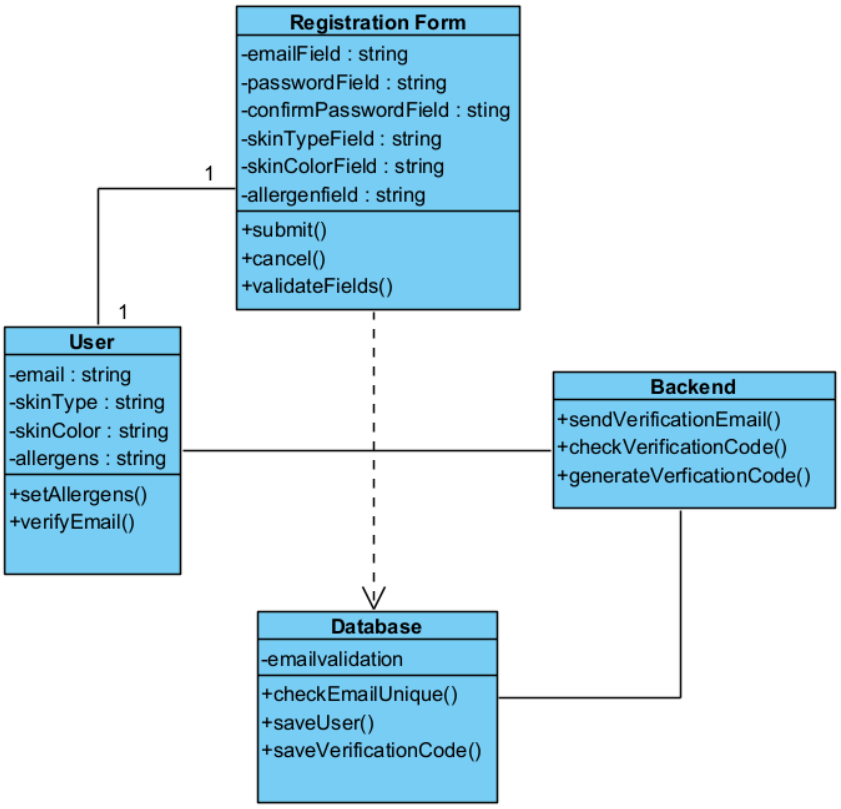
The system imposes a Restriction on Code Requests to prevent misuse. If the user requests a new code more than twice within a 10-minute window, the system displays a message: “Please wait 10 minutes to request a new code.” This ensures the process remains secure and prevents spamming of verification codes.

Additionally, the system handles Multiple Incorrect Attempts with further restrictions. If the user enters the wrong code three times, the system prompts them with the message: “You’ve entered the wrong code 3 times, please request a new one.” This ensures users do not repeatedly attempt incorrect codes without resolving the issue.

The registration process is designed with clear Key Flow Points to ensure user guidance at every step. In the Success Path, users are successfully registered and redirected to the home page upon entering the correct code. In cases of errors,

the system provides feedback for duplicate email detection, incorrect code entry, and limits on code requests or incorrect attempts. At each decision point, users are given clear options to correct their actions or proceed with the process, ensuring a smooth and user-friendly experience.

Class Diagram:



This diagram outlines the relationships between various classes and their methods, attributes, and functionalities, providing a detailed view of the system architecture.

The Registration Form class serves as the user interface where individuals input their registration details. Its attributes include fields such as emailField, passwordField, and confirmPasswordField for user credentials, along with skinTypeField, skinColorField, and allergenField for additional personal information. The class includes methods like submit() for sending data for processing, cancel() to abort the registration, and validateFields() to ensure all input fields meet the required criteria, such as proper email format and password match.

The User class represents the individual registering on the platform. It holds attributes like email, skinType, skinColor, and allergens to capture user-specific information. Key methods in this class include setAllergens(), allowing users to specify allergens, and verifyEmail(), which facilitates email verification by checking the confirmation code sent to their inbox.

The Backend class handles the logic and operations behind the scenes. It includes methods such as sendVerificationEmail() to deliver confirmation emails, checkVerificationCode() to validate the code entered by the user, and generateVerificationCode() to create unique codes for email confirmation.

The Database class is responsible for managing data storage and retrieval. It offers methods like emailValidation() to verify the email format and existence, checkEmailUnique() to ensure the email is not already registered, saveUser() to store user details, and saveVerificationCode() to record generated codes for comparison during email verification.

The relationships among these classes define their interactions. The User and Registration Form classes share a one-to-one relationship, where the user inputs data through the form. The Registration Form relies on the Backend to process data, send verification emails, and validate codes. Meanwhile, the Backend interacts with the Database to perform tasks like email validation, saving user information, and storing verification codes.

The System Workflow begins with user input, where the registration form collects and validates the data. Next, the backend checks email uniqueness and verification codes by interacting with the database. The email verification process involves sending a confirmation email through the backend, and the user must input the code for verification. Once all validations are successfully completed, the user data is stored in the database, completing the registration process.

2.3.Log In

The login form allows users to access their accounts with their email and password.

Login Form:

- Email*
- Password*
- Login, Forgot Password

Login:

- If the entered email is not registered/doesn't match or the entered password is incorrect displays an error message: "*Login information is not correct. Try again.*" If no errors occur, the user logs in.

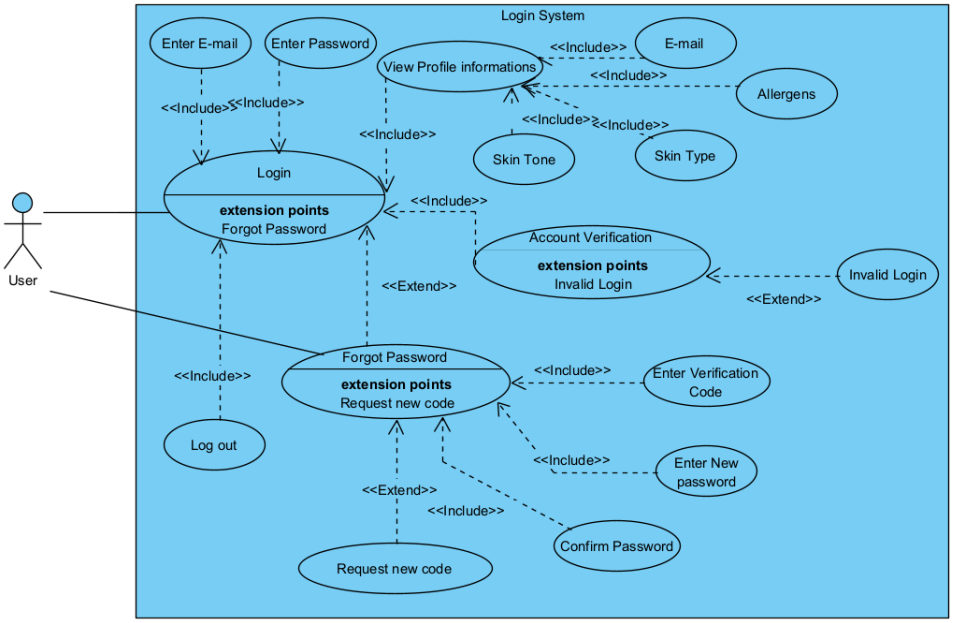
Forgot Password:

- The system sends a 4-digit confirmation code to the user's email that is entered in the login form if mail is not entered an error message is displayed: "Please enter your email address first.*"*. If the entered email is not registered, an error message is displayed: *"Email is not registered."* The user must enter the code to proceed, and if the entered code does not match the sent code, an error message is displayed: *"Confirmation code is incorrect."* Users also have the option to request a new code if necessary. However, to prevent misuse: If a user requests a new code more than twice within 10 minutes, the system displays the message: "*Please wait 10 minutes before requesting a new code.*" If the user enters the wrong code three times, the system displays the message: "*You have entered the wrong code three times. Please request a new code.*"Once the correct code is submitted, the user is allowed to reset their password. If the new password and its confirmation do not match, an error message is shown: *"Passwords don't match."* When the new password is successfully saved, a confirmation message is displayed: *"Your new password is saved."* Redirects the user to the home page.
- Checks if the password includes at least one uppercase letter, one lowercase letter, one number, and one special character; otherwise, displays an error message: "*Password must contain at least one uppercase letter, one lowercase letter, one number, and one special character.*"

Log Out:

3. Users can log out using the "LogOut" button.
4. Upon clicking the "LogOut" button, the system terminates the user's session and redirects them to the main page.

Use Case Diagram



This use case diagram presents a comprehensive overview of the interactions between the user and the system, emphasizing core functionalities such as logging in, resetting a password, viewing profile information, and logging out.

The primary actor in the system is the User, who performs all key operations within the system. These operations include logging in, resetting a forgotten password, viewing profile details, and securely logging out, which collectively shape the user's overall experience.

The Login process is a fundamental use case where the user must enter their email and password to access the system. This process includes mandatory steps, such as entering an email and password, represented by the "Include" relationship. Additionally, it features optional extension points like "Forgot Password," which allows users to recover their account when they forget their password, and "Invalid Login," which handles incorrect login attempts by displaying an error message. The Login process can also include "Account Verification" to ensure the user's identity.

The Forgot Password process is a critical recovery flow that helps users regain access to their accounts. This process includes steps like entering a verification code received via email and creating a new password. The system verifies the correctness of the verification code and ensures that the new password is confirmed correctly. Users also have the option to request a new verification code through the "Request New Code" extension point if needed.

The Account Verification process ensures that the user's account is valid during login. If verification fails, the system triggers the "Invalid Login" scenario, guiding the user with appropriate feedback and possible recovery actions.

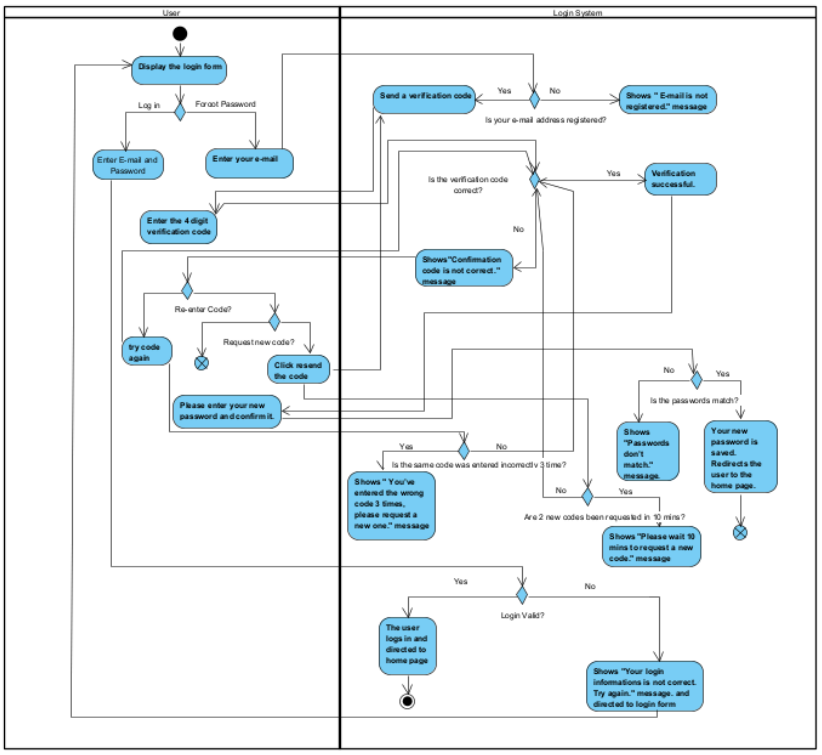
The View Profile Information use case enables users to access their personal details after successfully logging in. This includes viewing their skin tone, skin type, allergens, and email address. These details are grouped under the "Include" relationship, reflecting their necessity within the profile viewing process.

The Log Out functionality allows users to securely exit the system, ending their session and ensuring a safe return to the main page or login screen.

The relationships between these use cases are carefully structured using "Include" and "Extend" associations. The "Include" relationship signifies essential steps, like entering login credentials or verification codes, while the "Extend" relationship introduces flexibility by providing optional paths, such as handling invalid logins or requesting new verification codes during password recovery.

In conclusion, this use case diagram offers a structured and user-friendly representation of the system's core operations, covering login, password recovery, profile viewing, and logout functionalities. It effectively balances mandatory actions with flexible extensions, ensuring a smooth and efficient user experience.

Activity Diagram



This flowchart provides a detailed breakdown of a login system process, clearly separating user actions and system responses into two swimlanes: User and Login System. The flowchart ensures a structured pathway for users to either log in or recover their accounts, emphasizing feedback and error handling throughout the process.

The process begins with the system displaying the Login Form to the user. At this point, the user is presented with two primary options: they can proceed to log in by entering their credentials or choose the "Forgot Password" option to initiate the password recovery process. This initial step serves as the gateway to the two main pathways of the system.

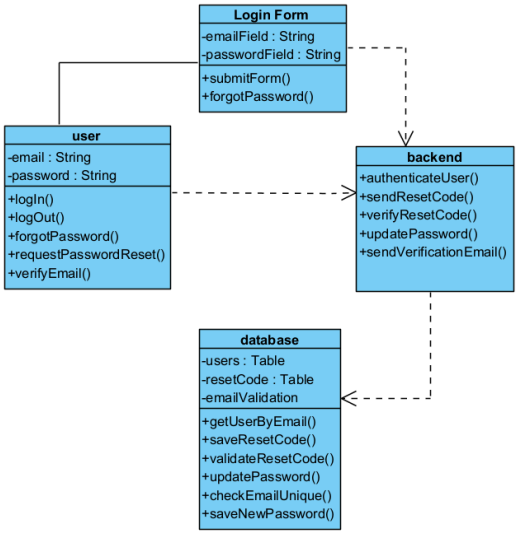
In the Login Process, the user inputs their email and password into the respective fields. The system then checks the validity of the entered credentials. If the login is valid, the user is successfully logged in and redirected to the home page. However, if the credentials are incorrect, the system displays the error message, "Your login information is not correct. Try again." This feedback ensures that users are aware of the issue and can attempt to resolve it.

If the user opts for the Forgot Password Process, they begin by requesting a verification code. The system sends a code to the user's registered email address. At this stage, the system checks whether the provided email address is registered. If it is not, the system displays the message, "Email is not registered." If the email is valid, the user proceeds to enter the 4-digit verification code received via email. The system verifies the code's correctness, and if it is incorrect, the user is informed with the message, "Confirmation code is not correct." Users are then given options to either re-enter the code or request a new one. To prevent abuse, the system imposes a limit: if the same code is entered incorrectly three times, the message "You've entered the wrong code 3 times. Please request a new one" is displayed. Additionally, if two codes are requested within a 10-minute window, the system prompts the user with "Please wait 10 minutes to request a new code."
Following successful verification, the user proceeds to the Password Reset Process. They are prompted to enter and confirm a new password. The system then checks whether the entered passwords match. If they do, the new password is saved, and the user is redirected to the home page. If the passwords do not match, the system displays the message, "Passwords don't match," prompting the user to try again.
Finally, the Login Completion step occurs once the user successfully logs in with valid credentials or completes the password reset process. In either case, the user is redirected to the home page, completing the flow.

Throughout the process, various Error Messages provide essential feedback to guide the user and ensure a smooth experience. These messages include:

"Email is not registered."
"Confirmation code is not correct."
"You've entered the wrong code 3 times. Please request a new one."
"Passwords don't match."
"Please wait 10 minutes to request a new code."
"Your login information is not correct. Try again."
In summary, this flowchart effectively organizes the user's actions and the system's responses into a clear and structured pathway. By addressing potential errors and providing user-friendly feedback, it ensures that users can navigate the login and recovery processes efficiently while minimizing confusion.

Class Diagram



The class diagram for the login system illustrates a modular architecture where different components interact to provide authentication, password reset, and email verification functionalities. This design ensures a clear separation of concerns, promoting maintainability and scalability. Below is a detailed explanation of each class and their roles in the system.

The Login Form represents the user interface where users input their login credentials. It contains two key attributes: emailField, which stores the email entered by the user, and passwordField, which stores the corresponding password. The form provides methods like submitForm() to send the entered credentials to the backend for authentication, and forgotPassword() to redirect the user to the password reset process.

The User class models the individual interacting with the system. It has attributes such as email and password, representing the user's credentials. Its methods include logIn() to authenticate the user, logOut() to end the session, and forgotPassword() to trigger the password reset process. Additional methods like requestPasswordReset() and verifyEmail() manage password recovery and email verification tasks.

The Backend class handles the core business logic of the system, coordinating authentication and password reset workflows. Key methods include authenticateUser() to validate login credentials, sendResetCode() to initiate the password recovery process by sending a reset code, and verifyResetCode() to confirm the user-provided code. Other methods like updatePassword() and sendVerificationEmail() support password updates and email validation.

The Database serves as the system's storage mechanism, maintaining user data and reset codes. It has two primary tables: users, which stores user emails and passwords, and resetCode, which temporarily holds reset codes for password recovery. The database provides methods such as getUserByEmail() to fetch user details, saveResetCode() to store reset codes, and validateResetCode() to ensure the code's validity. Additionally, methods like updatePassword() and checkEmailUnique() ensure data consistency and prevent duplicate email registrations.

The relationships between these components define the system's workflow. The Login Form interacts with the User class to handle login and password reset requests. The User communicates with the Backend for tasks like authentication, password recovery, and email verification. The Backend relies on the Database to fetch and update user data, manage reset codes, and validate email uniqueness.

In summary, this login system architecture is structured around three main layers: the Frontend (Login Form) manages user interactions, the Backend processes requests and handles logic, and the Database ensures data persistence and consistency. These components work together seamlessly, supporting essential features like user authentication, password reset, and email verification, making the system both efficient and robust.

2.4. Update Profile

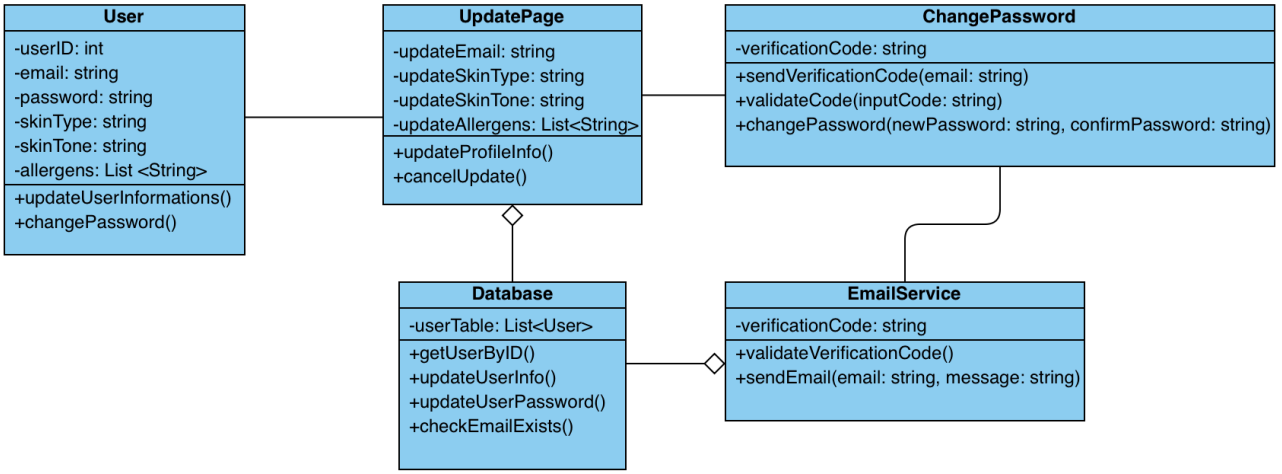
Users can update their profile information, including:

- **Email**
  Users can change their email address by entering a new one in the textbox. If the entered email address is already in the database, an error message will be displayed: "This email address is already registered."
- **Skin Type and Skin Tone**
  Users can select a new skin type or skin tone from a dropdown list that appears when clicking on the respective fields.
- **Allergens**
  Users can update their allergens in the textbox. If the input is not in the correct format, the system displays an error message:
  "Please enter allergens in the desired form, such as Paraben, Alcohol, etc."

Update: All updated information is saved to the database if there are no errors.
Cancel: Reverts changes.

Change Password:

- A 4-digit confirmation code is sent to the user's email. The user must enter the code to proceed. If the code is incorrect, an error message appears: *"Confirmation code is incorrect."*
- Users can request a new code if needed. However, the procedures that the system has when sending a verification code also apply to this stage.
- Checks if the password includes at least one uppercase letter, one lowercase letter, one number, and one special character; otherwise, displays an error message: "*Password must contain at least one uppercase letter, one lowercase letter, one number, and one special character.*"
- Once the correct code is entered, the user can set a new password. If the new password and confirmation do not match, the system shows an error: *"Passwords don't match."*
- When successfully updated, a message is displayed: *"Your new password is saved."* Redirects the user to the home page while the user stays logged in.



Class Diagram

This Class Diagram depicts the architecture and interaction of a user profile management system designed to handle user information updates, password changes, and email-based verification. The system ensures secure and efficient management of user data with functionalities like profile updates and password recovery.

Classes and Attributes:

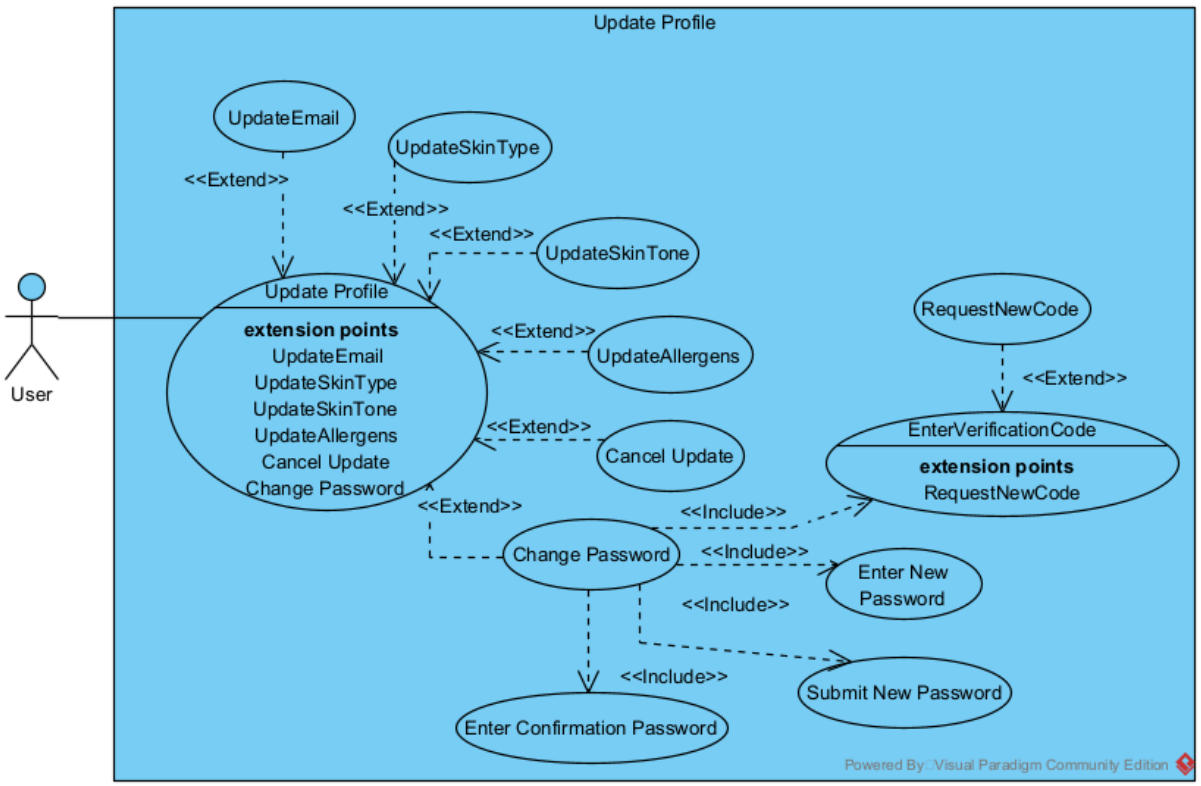
1. **User**
   - **Attributes:**
     - userID: int – Unique identifier for the user.
     - email: string – Email address of the user.
     - password: string – User's login password.
     - skinType: string – User's skin type (e.g., oily, dry, sensitive).
     - skinTone: string – User's skin tone for personalized recommendations.
     - allergens: List<String> – A list of allergens to avoid in products.
   - **Methods:**
     - updateUserInformations() – Updates the user's personal information.
     - changePassword() – Initiates the password change process.
2. **UpdatePage**
   - **Attributes:**
     - updateEmail: string – New email address to update.
     - updateSkinType: string – New skin type to update.
     - updateSkinTone: string – New skin tone to update.
     - updateAllergens: List<String> – New list of allergens to update.
   - **Methods:**
     - updateProfileInfo() – Saves the updated profile information.
     - cancelUpdate() – Cancels the ongoing update process.

3. **Database**
    - **Attributes:**
        - userTable: List<User> – A collection of all user profiles.
    - **Methods:**
        - getUserByID() – Retrieves user details using their ID.
        - updateUserInfo() – Updates user details in the database.
        - updateUserPassword() – Updates the user's password in the database.
        - checkEmailExists()- Checks if the email already exists.
4. **ChangePassword**
    - **Attributes:**
        - verificationCode: string – Code used for email verification.
    - **Methods:**
        - sendVerificationCode(email: string) – Sends a verification code to the specified email address.
        - validateCode(inputCode: string) – Validates the entered verification code.
        - changePassword(newPassword: string, confirmPassword: string) – Changes the user's password upon successful validation.
5. **EmailService**
    - **Attributes:**
        - verificationCode: string – A generated code for email verification.
    - **Methods:**
        - validateVerificationCode() – Verifies the correctness of the sent code.
        - sendEmail(email: string, message: string) – Sends an email to the specified address with a custom message.

Class Relationships:

- The **User** class connects to the **UpdatePage** class, enabling users to update their profile information.
- The **Database** class serves as the central repository, storing and updating user data.
- The **ChangePassword** class works in conjunction with **EmailService** to handle secure password changes, including email-based verification.
- The **EmailService** class facilitates email communication, including sending verification codes and validating them for password recovery or updates.

Use Case Diagram:



The Update Profile use case allows the user to update various attributes of their profile. The following functionalities are supported as extension points:

1. **UpdateEmail:**
    - Allows the user to update their email address.
    - Requires email verification for security.
    - Extends the Update Profile use case.
2. **UpdateSkinType:**
    - Enables the user to modify their skin type information.
    - Extends the Update Profile use case.
3. **UpdateSkinTone:**
    - Allows users to update their skin tone information.
    - Extends the Update Profile use case.
4. **UpdateAllergens:**
    - Allows users to update any allergen information in their profile.
    - Extends the Update Profile use case.
5. **Cancel Update:**
    - Lets the user cancel the profile update process at any point.
    - Extends the Update Profile use case.
6. **Change Password:**
    - Provides the functionality to change the user's password.
    - Includes sub-processes such as entering a confirmation password and submitting a new password.

Sub-Use Cases of Change Password

1. **Enter Confirmation Password:**
    - Users must enter their current password to confirm their identity before changing their password.
    - Included in the Change Password use case.
2. **Enter New Password:**
    - Users input their desired new password.
    - Included in the Change Password use case.
3. **Submit New Password:**
    - Finalizes the password change process by saving the new password.
    - Included in the Change Password use case.

Additional Use Case: EnterVerificationCode

1. **RequestNewCode:**
    - If the verification code is not received or has expired, the user can request a new code.
    - Extends the EnterVerificationCode use case.
2. **EnterVerificationCode:**
    - Handles the process of inputting the verification code required to validate actions like updating the email address.
    - Provides secure verification as an extension point.

Key Relationships

- **Extend:**
    - The Update Profile use case is extended by specialized update functionalities like updating email, skin type, skin tone, allergens, and canceling updates.
    - Similarly, EnterVerificationCode is extended by the option to request a new code if necessary.
- **Include:**
    - The Change Password use case includes smaller steps, such as entering a confirmation password, inputting the new password, and submitting it for finalization.

2.5. Skin Type Test

The Skin Type Test is a core component of GlowGenie that allows users to accurately determine their skin type. This feature is essential for users who are unsure of their skin type or want to validate their knowledge. The process uses machine learning to analyze user responses and classify skin types into the following categories:

- **Dry**
- **Oily**
- **Combination**
- **Normal**

HomePage: Directs the user to the home page.

Submit: Submits the test to be evaluated and saved to the database.

Input

User Responses:

- Users answer a series of multiple-choice questions regarding how their skin behaves under different conditions. These questions include:

1. **How often does your skin feel oily or shiny, especially in the T-zone (forehead, nose, and chin)?**

   A. Always

   B. Rarely

   C. Sometimes

   D. Frequently

2. **Does your skin feel tight, dry, or flaky after cleansing?**

   A. Always

   B. Often

   C. Occasionally

   D. Never

3. **How do moisturizers make your skin feel?**

   A. Feel perfectly hydrated

   B. Feel greasy and sticky

   C. Feel slightly oily in some areas but fine in others

   D. Absorb quickly without feeling greasy

4. **Do you often experience blackheads or acne, especially in your T-zone (forehead, nose, chin)?**

   A. Always

   B. Frequently

   C. Occasionally

   D. Never

5. **How does your skin feel after waking up in the morning?**

A. Feels perfectly balanced

B. Feels dry or tight

C. Feels normal in some areas but oily in the T-zone

D. Feels greasy or shiny all over

6. **How does your skin react to seasonal changes?**

   A. No noticeable change

   B. Feels drier in winter, normal in summer

   C. Feels normal in winter, oily in summer

   D. Feels oily or greasy regardless of season

7. **Does your skin feel comfortable and balanced throughout the day, without the need for extra moisturizers or products?**

   A. Always

   B. Occasionally

   C. Rarely

   D. Never

8. **How does your U-zone (cheeks and jawline) typically feel?**

   A. Very oily throughout the day

   B. Slightly oily by the end of the day

   C. Normal, neither oily nor dry

   D. Dry or tight most of the time

- The form allows users to select one answer per question from predefined options.
- The questions are designed to cover various aspects of skin characteristics, ensuring a comprehensive evaluation.

Mandatory Fields:

- All questions in the Skin Type Test are mandatory. If users attempt to submit the form without completing all required fields, the system will display the following error message **"***Please fill in the mandatory areas.***" a**nd won't proceed to save answers unless all questions are answered.

Processing

1. **Data Collection:**
   - User responses are collected and preprocessed to ensure completeness.
2. **Feature Extraction:**
   - Each question maps to specific features (e.g., high oil production maps to "Oily," low hydration to "Dry").
3. **Machine Learning Model:**
   - A trained classification model (SVM) analyzes the responses.
   - The model is trained on labeled datasets containing skin type information derived from user surveys.
   - Based on the answers, the model predicts the most likely skin type for the user.
4. **Model Output:**

    - The model returns one of the four skin types: Dry, Oily, Combination, or Normal.
5. **Profile Update:**
    - Once the skin type is determined, it is automatically saved to the user's profile.

Output

- The identified skin type is displayed to the user immediately after submission.

Use Case Diagram

This Use Case Diagram represents the user interactions and underlying workflows of a skin type detection application. The diagram features the **User** actor, interacting with various use cases through **Include** and **Extend** relationships, illustrating how the system responds to different scenarios.

Diagram Components:

1. **Actor:**
    - **User:** The individual who interacts with the system and performs the test.
2. **Use Cases:**
    - **Take Test:** The user initiates the skin type test.
    - **Answer Questions:** The user answers questions related to their skin condition during the test.
    - **Submit Test:** The user submits their responses after completing the test.
    - **View Test Result:** After submission, the user views their skin type result.
    - **Update User Profile:** The user's profile is updated based on the test results.
    - **Handle Error:** If there are missing fields, the system notifies the user and prompts correction.

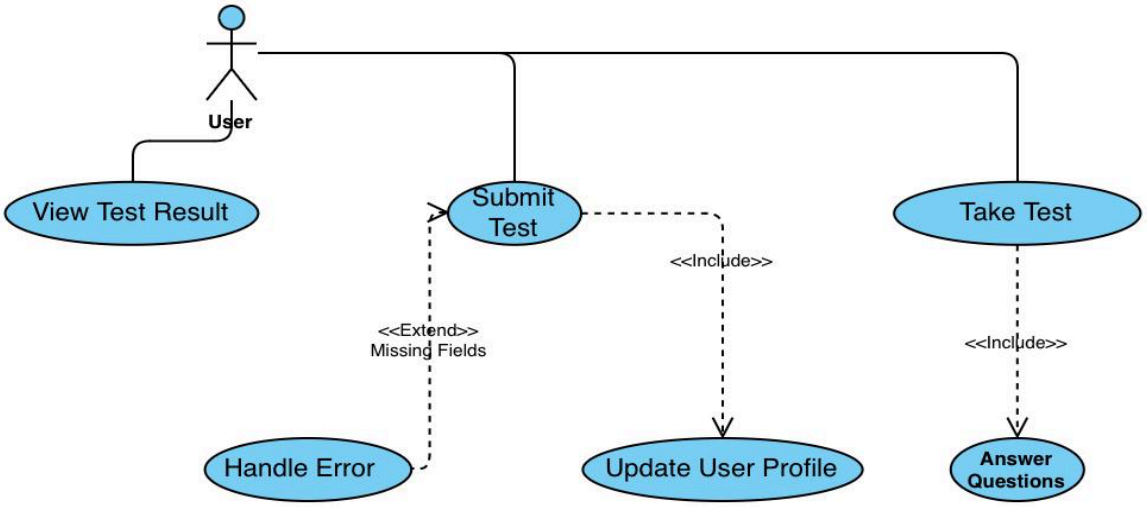
Diagram Relationships:

1. Include Relationship:

- The **"Take Test"** process **includes** the **"Answer Questions"** use case.
    - **Reason:** For a test to be completed, the user must answer the questions. This step is **mandatory** and occurs as an **integral part** of every test without exception.
    - **Technical Explanation:** The "Answer Questions" use case is **essential** for the "Take Test" process to proceed. The test cannot advance if the questions are not answered.
- The **"Submit Test"** use case **includes** the **"Update User Profile"** use case.
    - **Reason:** When the test is submitted, the user's profile is automatically updated. This process occurs **sequentially** and does not require direct user intervention.
    - **Technical Explanation:** The "Update User Profile" use case is **an inherent part** of the "Submit Test" process. Once the test data is submitted, the profile update **inevitably** takes place.

2. Extend Relationship:

- The **"Submit Test"** use case is **extended** by the **"Handle Error"** use case.
    - **Reason:** When the user submits the test, the system automatically checks for missing or incorrect fields. If any are detected, the **"Handle Error"** process is triggered.
    - **Technical Explanation:** The "Handle Error" use case is **conditional** and activates **only if** missing responses or errors are found during test submission. If the test is submitted without errors, the "Handle Error" process is **skipped**, and the main flow continues uninterrupted.
    - **Decision Point:** The "Submit Test" process contains a **decision point**, and the "Handle Error" use case operates as an **alternate flow** triggered by error detection.



Activity Diagram

This Activity Diagram represents the process of a skin type detection test, from the initiation by the user to the storage of results in the database. The diagram consists of five components (swimlanes): User, Frontend, Backend, Machine Learning (ML Model), and Database.

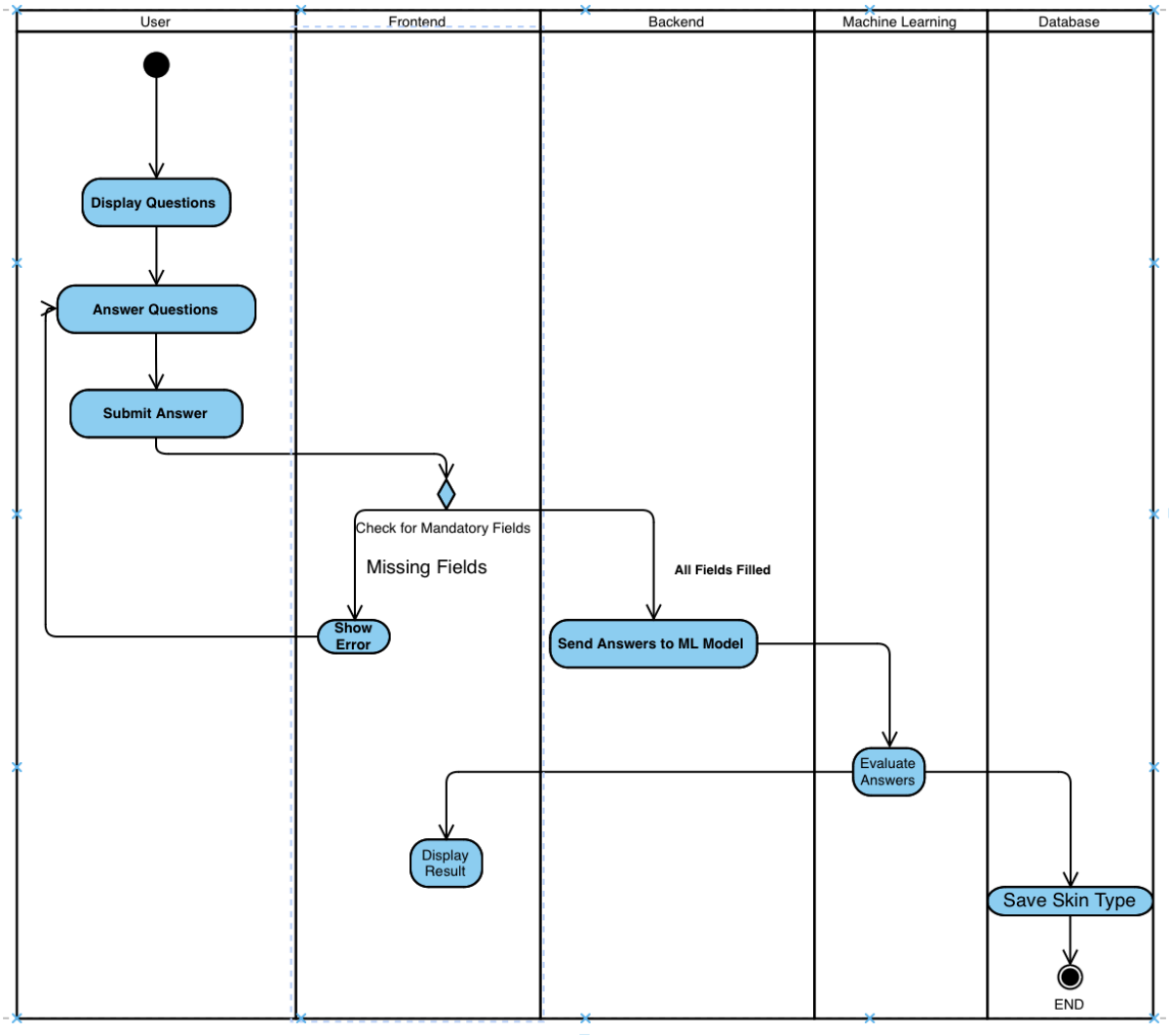
Diagram Components and Workflow:

1. **Start:**
   - The user initiates the process.
2. **Display Questions:**
   - The system displays test questions to the user. This action takes place in the User swimlane.
3. **Answer Questions:**
   - The user answers the questions.
4. **Submit Answer:**
   - The user submits their responses, transitioning the process to the Frontend layer.
5. **Check for Mandatory Fields:**
   - The Frontend checks if all mandatory fields are filled.
   - If fields are missing, the process proceeds to the Show Error step, requiring the user to complete the missing fields.
   - If no fields are missing, the process moves to the Backend.
6. **Send Answers to ML Model:**
   - The Backend sends the responses to the Machine Learning Model for analysis.
7. **Evaluate Answers:**
   - The ML Model evaluates the responses and determines the user's skin type.
8. **Display Result:**
   - The detected skin type is sent to the Frontend and displayed to the user.
9. **Save Skin Type:**
   - The identified skin type is saved in the Database through the Backend. This is the final step of the process.
10. **End:**
    - The process concludes once the skin type is stored in the database.

Technical Points:

- **Decision Node:**
  - Mandatory fields are checked. If any fields are missing, the user is prompted with an error message. Otherwise, the process continues smoothly.
- **Machine Learning Integration:**
  - Responses are passed from the backend to the Machine Learning (ML) Model for skin type evaluation.
- **Database Update:**

- The skin type result is stored in the database through the Save Skin Type step.



Class Diagram

This Class Diagram illustrates the core components and relationships within a skin type detection system. The diagram represents the flow where a user completes a form, submits answers, and the machine learning model processes these answers to predict the skin type, updating the user's profile with the results.

Classes and Attributes:

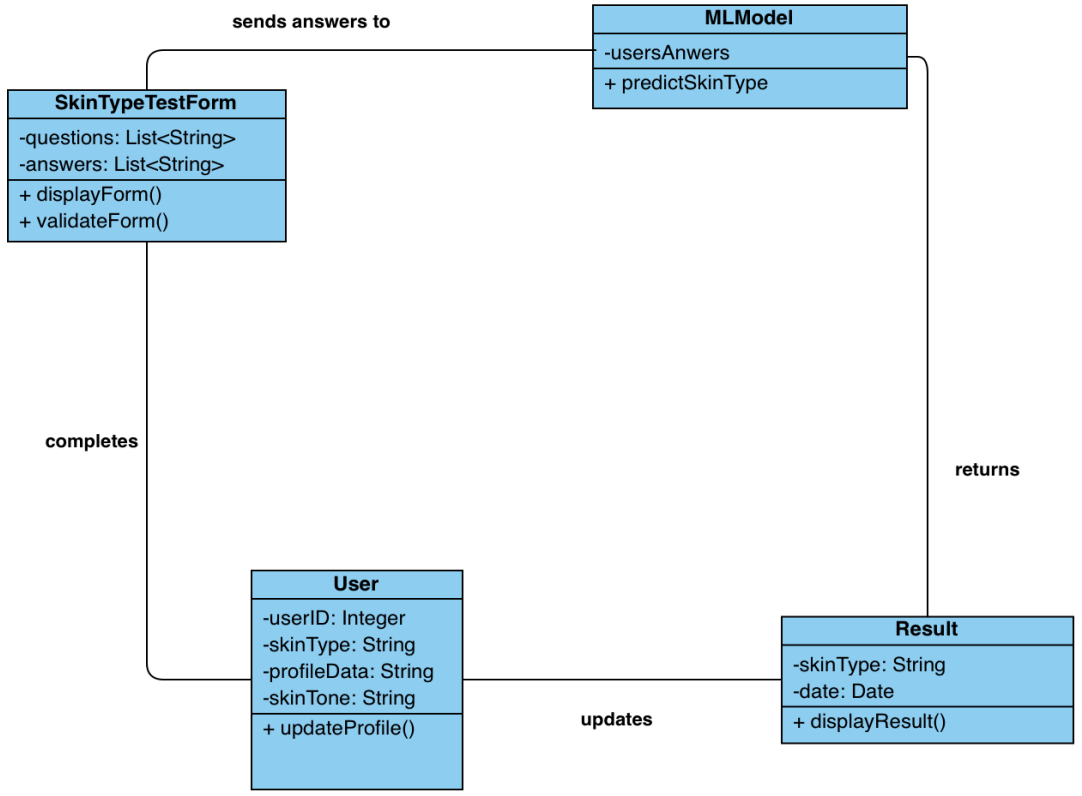
1. **SkinTypeTestForm:**
   - **Attributes:**
     - questions: List<String> – A list of questions presented to the user.
     - answers: List<String> – A list of responses provided by the user.
   - **Methods:**
     - displayForm() – Displays the test form to the user.
     - validateForm() – Validates user responses for completeness and accuracy.
2. **MLModel:**
   - **Attributes:**
     - usersAnswers – Answers submitted by the user.
   - **Methods:**
     - predictSkinType() – Analyzes responses and predicts the user's skin type.
3. **Result:**
   - **Attributes:**
     - skinType: String – The detected skin type.
     - date: Date – The date the test was conducted.

- **Methods:**
  - displayResult() – Displays the test results to the user.

4. **User:**
   - **Attributes:**
     - userID: Integer – Unique identifier for the user.
     - skinType: String – The user's skin type.
     - profileData: String – Additional profile information.
     - skinTone: String – The user's skin tone.
   - **Methods:**
     - updateProfile() – Updates the user's profile based on test results.

Class Relationships:

- The **SkinTypeTestForm** class enables the user to complete the skin type test (**completes**).
- Answers are sent to the **MLModel** class (**sends answers to**), where they are analyzed, and predictions are made.
- The **MLModel** returns results to the **Result** class (**returns**).
- The **Result** class updates the **User** class with the detected skin type (**updates**).



2.6. Product Suitability Feedback

This feature enables users to evaluate whether a specific product is compatible with their skin type. It is implemented using a combination of machine learning and GPT API integration, ensuring personalized and accurate recommendations.

Input:

If skinType = "I don't know", display a pop-up when the user attempts to access the suitability feature.

- Navigate to the home page to update their skin type.
- Navigate to take the Skin Type Test.

HomePage: Directs the user to the home page.

Processing Steps

1. Data Retrieval

The system searches a product database or external APIs to fetch details about the entered product. The system searches products in the database according to the user's profile information (skin type, skin tone, allergens if entered). It shows whether the product the user enters is suitable for their skin type. These include:

- **Ingredient List**: The core components of the product.
- **Concentration Levels**: Ingredient proportions, which are important for determining their impact.
- **Category**: The category of the product (must match one of the five predefined categories: *moisturizer, toner, sunscreen, cleanser, serum*).
    - If the product is a **sunscreen**, the system also retrieves information about which **skin tones** it is suitable for.

2. Skin Type Labeling Model

- This model is responsible for assigning a skin type label to a product based on its ingredients.

Purpose:

- To classify newly queried products and label them with the skin type(s) they are suitable for.

Input:

- **Features**
    - Ingredients List
    - Product Category
    - Skin Type It's Suitable for (to filter for sunscreen).

Output

- **Skin Type Label:**

  A classification label (e.g., "Oily", "Dry", "Normal", "Combination").

Model Training:

- Random forest is going to be used to train the model.

3.Ingredient Contribution Analysis Model

- This model is responsible for assigning a skin type label to a product based on its ingredients.

Purpose:

- To analyze and give feedback on which ingredients list the chosen product being classified as suitable or unsuitable for a specific skin type.

Input:

Features:

- Ingredient List(same as Skin Type Labeling Model).
- Product Category
- Skin type It's Suitable for (to filter sunscreen) output from Predicted skin type label.

Output:

- A list of ingredients contributing to suitability or unsuitability for a specific skin type.

Model Training:

- Random forest model paired with SHAP (Shapley Additive Explanations) to evaluate feature importance.

Ingredient Analysis:

- After Skin Type Labeling Model assigns a skin type label to a product, going to use SHAP values or feature importances to determine which ingredients contributed most to the classification

.

The activity diagram illustrates the step-by-step workflow for generating tailored product recommendations based on the user's skin type, skin tone, and allergens. The process includes five key components: **User**, **Frontend**, **Backend**, **Database**, and **ML Model**.

1. **User Interaction**:
    - The user clicks on the "Generate Products for Me" button.
    - If the user's skin type is marked as "I don't know," a pop-up is shown to prompt them to either update their skin type on the home page or take a skin type test.
2. **Frontend Actions**:
    - After the user updates or confirms their skin type, they proceed to choose product categories (e.g., cleanser, toner).
    - A filtered list of products from the selected categories is displayed based on user preferences.
3. **Backend Actions**:
    - The backend gathers necessary input (skin type, tone, allergens, and categories) to query the product database.
    - If no products satisfy the user's preferences for a category, an error message is displayed, listing the categories without suitable products.
4. **ML Model Integration**:
    - When the user selects a product, the ML model evaluates its ingredients to determine suitability for the user's skin type.
    - The ML model checks if the product has been previously evaluated for the same skin type.
    - Feedback is generated, explaining why the product is suitable or unsuitable based on its ingredients.
5. **Output**:
    - Suitable products are framed in **green**, and unsuitable products in **red**.
    - Users can view detailed feedback on product suitability.

Compatibility Result

The results are presented in two columns:

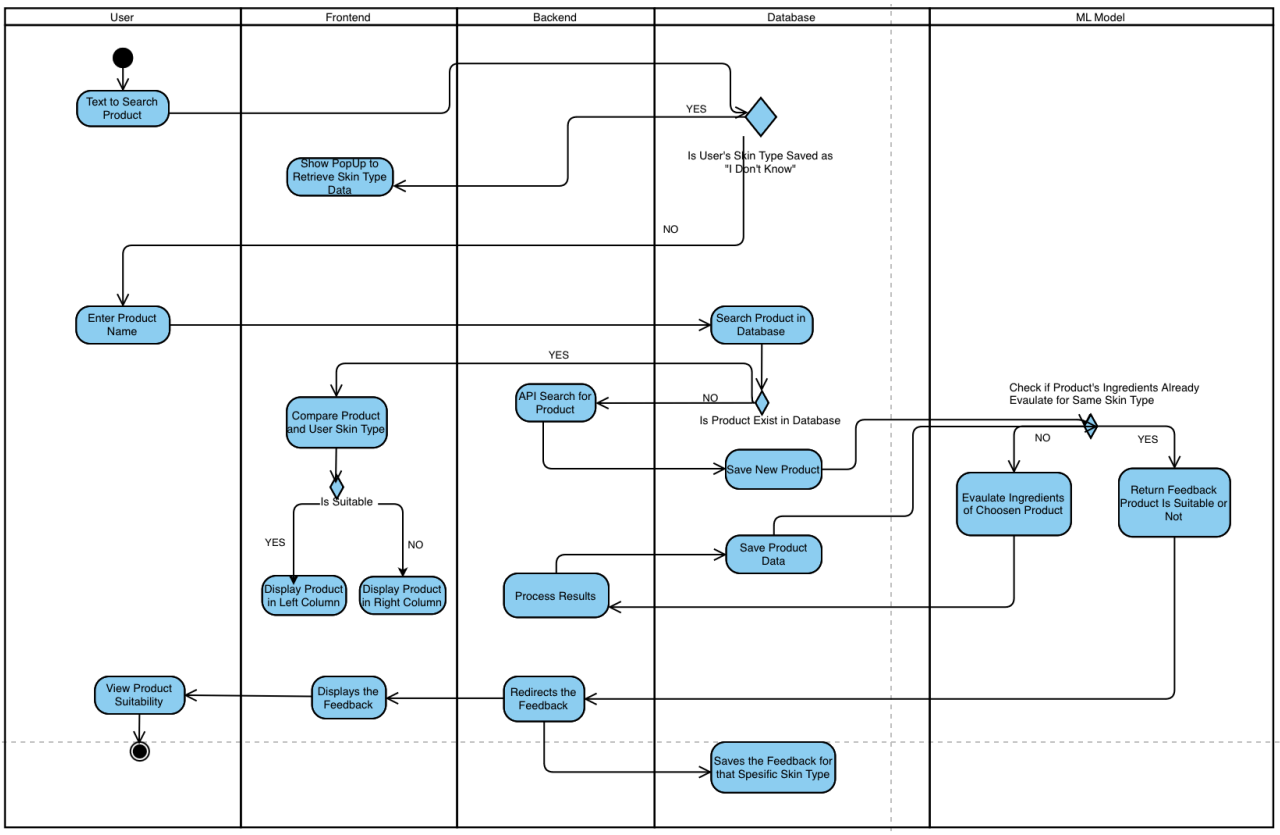
1. **Left Column**: Products suitable for the user's skin type.
2. **Right Column**: Products unsuitable for the user's skin type.

Each product is displayed in a **card/frame** format containing:

- **Green**: Suitable for the user.
- **Red**: Not suitable for the user.

Users can click on a product to view its **detailed information** in a pop-up, including:

- Product Name
- Ingredients
- Feedback that explains which specific ingredients in the product make it suitable or unsuitable for the user's skin type



Activity Diagram

This Activity Diagram illustrates the workflow of the Product Suitability Feedback feature, guiding the process from a user's product search to the final evaluation of product compatibility with their skin type. The diagram is divided into five swimlanes: **User**, **Frontend**, **Backend**, **Database**, and **ML Model**, representing different components involved in the process.

Diagram Components and Workflow:

1. **Start:**
   - The process begins when the user initiates a product search by entering text in the search bar.
2. **Retrieve Skin Type:**
   - If the user's skin type is saved as "I Don't Know," a pop-up appears prompting the user to take a skin type test or update their profile on the home page. This ensures the system has sufficient information to provide accurate recommendations.
3. **Enter Product Name:**
   - The user inputs the name of the product they wish to evaluate.
4. **Product Search:**
   - The backend performs a product search in the database.
   - If the product exists, the system proceeds to API calls to retrieve further details.
   - If the product is not found, it is saved as a new entry in the database along with relevant product data.

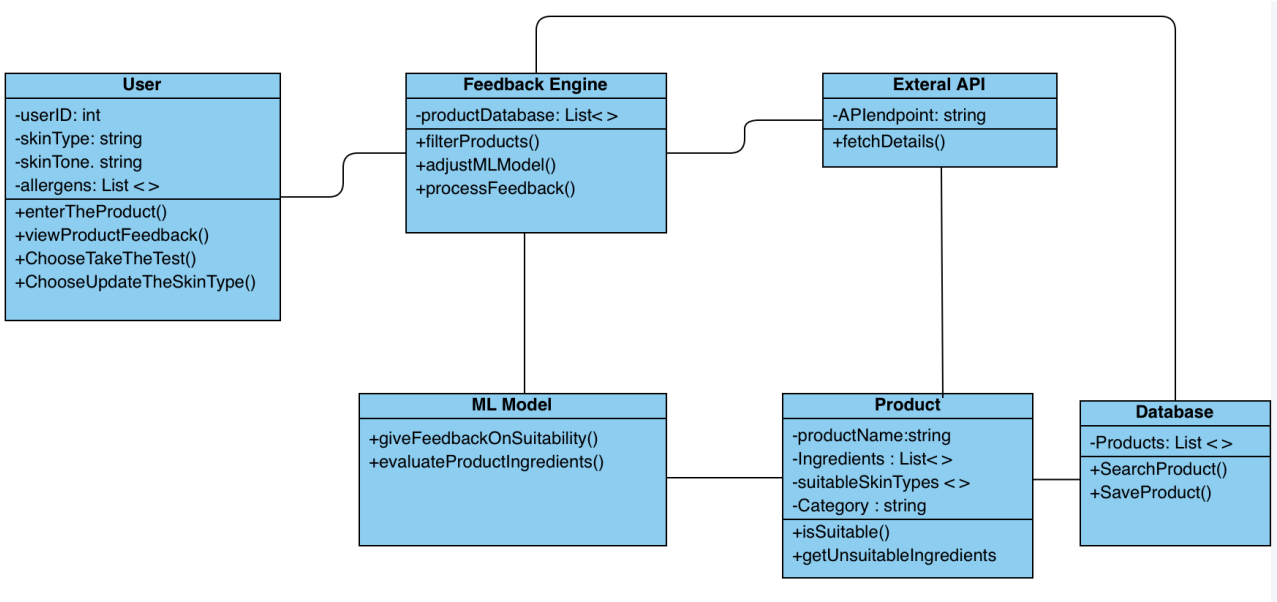
5. **Evaluate Product Ingredients::**
   - The system checks if the product's ingredients have already been evaluated for the user's skin type.
   - If the evaluation exists, the ML Model returns feedback immediately.If no fields are missing, the process moves to the Backend.
   - If not, the ML Model evaluates the product ingredients and classifies the product as either suitable or unsuitable.
6. **Comparison and Display::**
   - The frontend compares the evaluated product with the user's skin type.
   - Products deemed suitable are displayed in the left column, while unsuitable products appear in the right column.
7. **Feedback and Viewing:**

- The user can view the product suitability details, which include ingredient analysis and explanations.

8. **Save Feedback:**
   - The feedback, including product compatibility with the user's skin type, is saved in the database for future reference.
9. **End:**
   - The process concludes when the user views the product feedback, and the system saves the evaluation results..

Technical Points:

- **Decision Node:**
  - The system checks if the user's skin type is known. If unknown, the user must update their profile before proceeding.
  - The product existence check determines whether the product needs to be added to the database.
- **Machine Learning Integration:**
  - The ML Model evaluates the product's ingredient list using classification techniques and SHAP values to provide transparent ingredient contribution analysis.
- **Database Update:**
  - Product evaluations and feedback are stored in the database for recurring user queries, improving response times for future searches.



Class Diagram

This Class Diagram illustrates the architecture and interactions within the product suitability feedback system. The system evaluates skincare products based on user attributes (skin type, skin tone(for sunscreen) and allergens) and provides feedback using machine learning models and external product databases.

Classes and Attributes:

1. **User:**

    - **Attributes:**
        - userID: int – Unique identifier for the user.
        - skinType: string – User's skin type (e.g., oily, dry, sensitive).
        - skinTone: string – User's skin tone (for evaluating products like sunscreen).
        - allergens: List<> – List of allergens to avoid.
    - **Methods:**
        - enterTheProduct() – Allows the user to input a product name.
        - viewProductFeedback() – Displays the feedback on product compatibility.
        - ChooseTakeTheTest() – Directs the user to take a skin type test.
        - ChooseUpdateTheSkinType() – Updates user's skin type profile.
2. **Feedback Engine:**
    - **Attributes:**
        - productDatabase: List<> – A list of products available for feedback and analysis.
    - **Methods:**
        - filterProducts() – Filters products based on user attributes and queries.
        - adjustMLModel() – Adjusts machine learning parameters for better predictions.
        - processFeedback() – Processes and refines product suitability feedback.
3. **ML Model:**
    - **Methods:**
        - giveFeedbackOnSuitability() – Generates feedback on whether a product is suitable for the user.
        - evaluateProductIngredients() – Evaluates the ingredients of a product to determine its compatibility.
4. **Product:**
    - **Attributes:**
        - productName: string – The name of the skincare product.
        - Ingredients: List<> – A list of ingredients present in the product.
        - suitableSkinTypes<> – Skin types for which the product is suitable.
        - Category: string – Product category (e.g., moisturizer, sunscreen).
    - **Methods:**
        - isSuitable() – Determines if the product is suitable for the user's skin type.
        - getUnsuitableIngredients() – Lists ingredients that may cause issues for the user.
5. **External API:**
    - **Attributes:**
        - APIendpoint: string – The endpoint used to fetch external product details.
    - **Methods:**
        - fetchDetails() – Fetches product data from external databases.
6. **Database:**
    - **Attributes:**
        - Products: List<> – A list of stored products.
    - **Methods:**
        - SearchProduct() – Searches for products in the database.
        - SaveProduct() – Saves new product entries.

Class Relationships:

- The **User** class interacts with the **Feedback Engine** to input products and receive feedback.
- The **Feedback Engine** communicates with both the **ML Model** and the **External API** to filter, adjust, and process product data.
- The **ML Model** evaluates product suitability and returns results to the **Feedback Engine.**
- **Products** are stored and retrieved through the **Database**, and the **External API** can fetch additional product details if needed.

2.7. Generate Product Recommendations

The system provides product recommendations tailored to the user's skin type, skin tone (for sunscreen), and allergens, if specified. The process involves:

Input

- **Skin Type**: Known from user input or determined via the skin type test. If the user's skin type is saved as **"I don't know"**, a mandatory pop-up will appear before accessing the recommendation interface. This pop-up requires the user to either:
    - ➔ Redirects the user to the Home Page so they update their skin type.
    - ➔ Choose to take the **Skin Type Test**.
- **Skin Tone**: Entered during registration.
- **Allergens**: Optional, entered by the user in a textbox.
- **Categories**: Selected by the user from available options (moisturizer, toner, sunscreen, cleanser, serum).

Processing

- Products in the database are filtered and listed for the user's view according to their profile information (skin type, skin tone, allergens if entered.) and also the categories they've chosen. The user can view a filtered list of items. If the user selects a product, a pre-trained machine learning model (Ingredient Contribution Analysis Model from 2.7) analyzes its ingredients and provides feedback, explaining why the product is suitable or unsuitable for their skin type.
- Products are visually framed:
    - **Green**: Suitable for the user.
    - **Red**: Not suitable for the user.

Output

- A list of products under each selected category, including:
    - Product Name
    - Ingredients
    - Feedback that explains which specific ingredients in the product make it suitable or unsuitable for the user's skin type

Error Handling

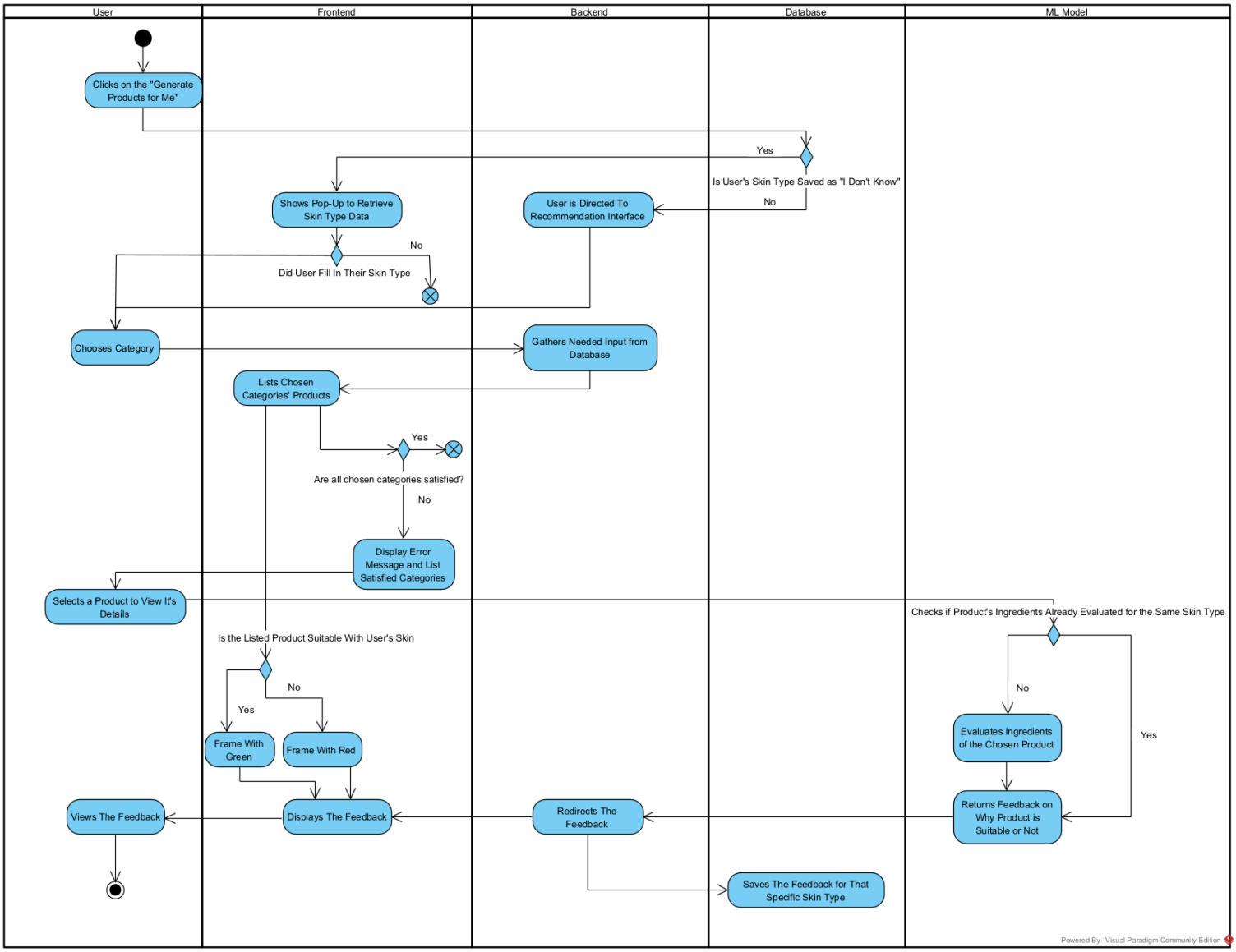
- At the start, the product database may not contain many entries. If no compatible product is found in the selected category or for the user's skin type, the system will return the following error message:
  "*There are no suitable products in [chosen_categories] category/categories for your skin type in our database at the moment. You can help us improve by querying products through our Suitability Service. This will add new products to our database and expand our recommendations over time.***"**

User Interface

- The user can:
    - View all products that are in the database, with suitability highlighted using the green/red framing system.
    - Select a product to view its detailed information including its ingredients and feedback given from the system.

HomePage: Directs the user to the home page.

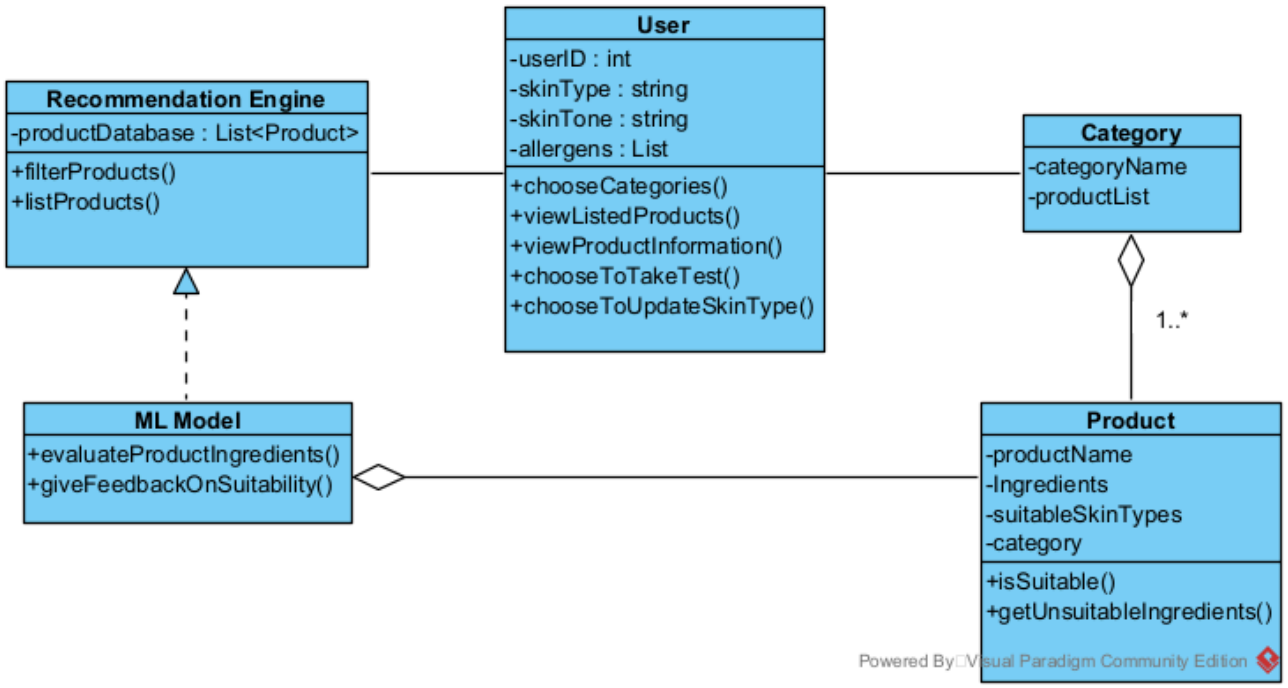
Activity Diagram: The activity diagram illustrates the step-by-step workflow for generating tailored product recommendations based on the user's skin type, skin tone, and allergens. The process includes five key components: **User**, **Frontend**, **Backend**, **Database**, and **ML Model**.



1. **User Interaction**:
   - The user clicks on the "Generate Products for Me" button.
   - If the user's skin type is marked as "I don't know," a pop-up is shown to prompt them to either update their skin type on the home page or take a skin type test.
2. **Frontend Actions**:
   - After the user updates or confirms their skin type, they proceed to choose product categories (e.g., cleanser, toner).
   - A filtered list of products from the selected categories is displayed based on user preferences.
3. **Backend Actions**:
   - The backend gathers necessary input (skin type, tone, allergens, and categories) to query the product database.
   - If no products satisfy the user's preferences for a category, an error message is displayed, listing the categories without suitable products.
4. **ML Model Integration**:
   - When the user selects a product, the ML model evaluates its ingredients to determine suitability for the user's skin type.
   - The ML model checks if the product has been previously evaluated for the same skin type.
   - Feedback is generated, explaining why the product is suitable or unsuitable based on its ingredients.
5. **Output**:

- Suitable products are framed in **green**, and unsuitable products in **red**.
- Users can view detailed feedback on product suitability.

Class Diagram:



The class diagram outlines the structure and relationships of the system components responsible for generating product recommendations.

1. **Classes**:
   - **User**:
     - Represents the user of the system, with attributes like userID, skinType, skinTone, and allergens.
     - Contains methods to choose categories, view listed products, view product details, take the skin type test, and update skin type.
   - **Category**:
     - Represents product categories (e.g., cleanser, moisturizer).
     - Contains attributes such as categoryName and productList.
     - Has a one-to-many relationship with the **Product** class.
   - **Product**:
     - Represents individual products in the database.
     - Attributes include productName, ingredients, suitableSkinTypes, and category.
     - Methods include checking suitability (isSuitable) and retrieving unsuitable ingredients (getUnsuitableIngredients).
   - **Recommendation Engine**:
     - Manages the product database and provides filtered product lists based on user preferences.
     - Contains methods for filtering and listing products.
   - **ML Model**:
     - Evaluates product ingredients to determine suitability using machine learning algorithms.
     - Provides detailed feedback explaining why a product is suitable or unsuitable.
2. **Relationships**:
   - The **User** interacts with the **Recommendation Engine** to filter and view products.
   - The **Recommendation Engine** relies on the **ML Model** to evaluate product ingredients and provide feedback.
   - Each **Category** is associated with multiple **Product** objects.
3. **Key Functionality**:
   - The system evaluates products for suitability based on user preferences (skin type, tone, allergens) using the recommendation engine and ML model.

- Feedback is saved in the database for future reference, ensuring continuous improvement of recommendations.

2.8. Update Ingredients of Skin Care Products'

The purpose of this function is to ensure that product data remains accurate and up-to-date by periodically refreshing ingredient information and re-evaluating product suitability. This enhances the reliability and relevance of recommendations provided to users.

Input

- Expiration Time: Each product in the database has an assigned expiration time for its ingredient data.
- Product ID: The unique identifier for the product whose ingredient data needs to be updated.

Process

Check Expiration:

- The system periodically checks all products in the database to determine if their expiration time has been reached.

Retrieve Updated Information:

- Query GPT to fetch the latest ingredient information for products whose data has expired.
- Replace the outdated ingredient list with the newly retrieved data.

Re-evaluate Product Suitability:

- Use the updated ingredient list as input to the machine learning model.
- Reclassify the product by determining its suitability for specific skin types.

Database Update:

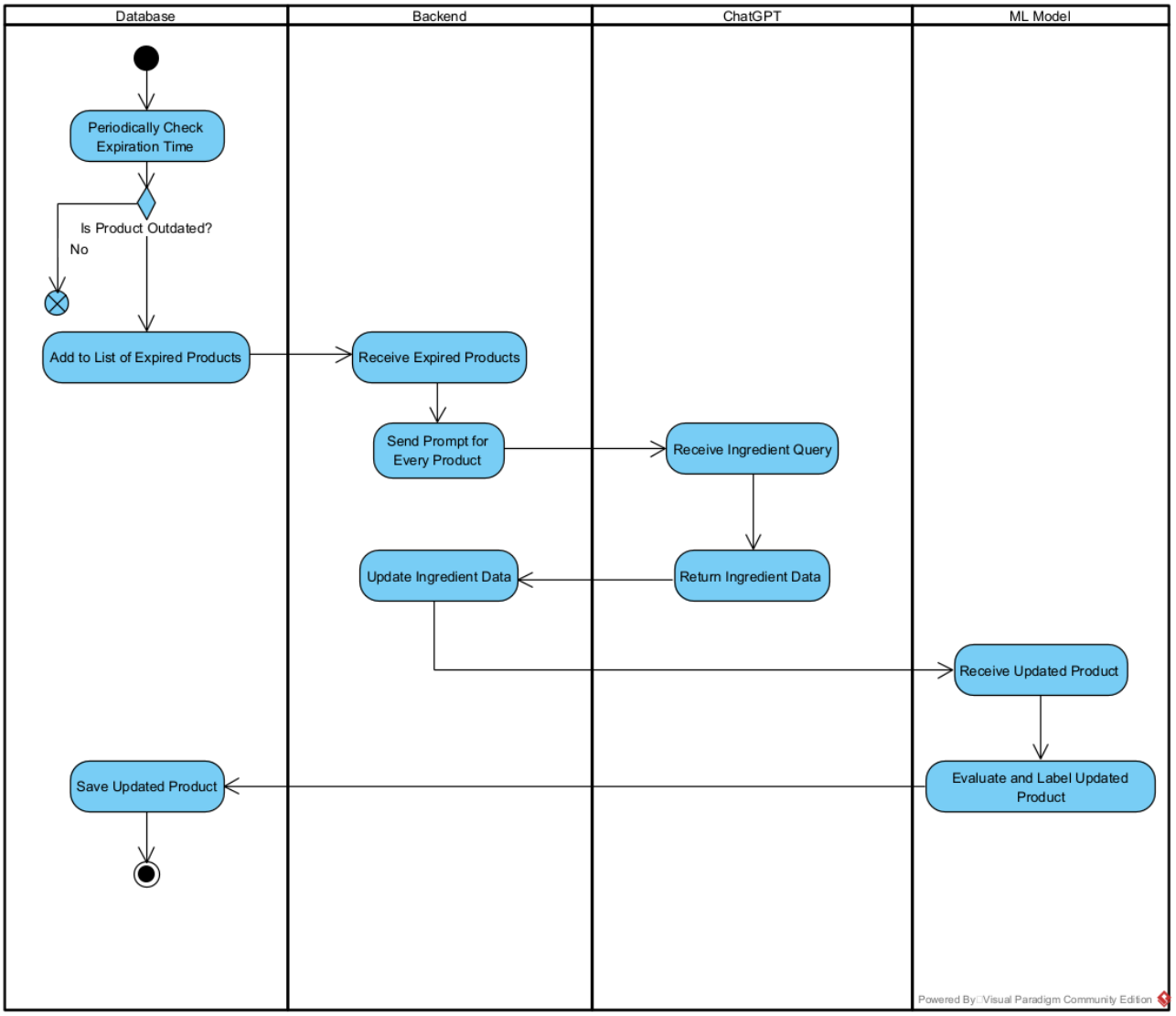
Replace the outdated product details in the database with the updated information, including the new suitability classification.

Output

- A refreshed and updated product database.
- Accurate suitability classifications for all products, ensuring high-quality recommendations.

Benefits

- Maintains the accuracy and relevance of the product database over time.
- Improves user trust by providing reliable recommendations based on up-to-date information.



Activity Diagram:

1. Database

1. Periodically Check Expiration Time:
   The database periodically checks if any product's expiration time has been reached.
2. Is Product Outdated?
   - If the product is not outdated, terminate the process for that product.
   - If outdated, add it to the list of expired products.
3. Add to List of Expired Products:
   The list of expired products is sent to the backend for further processing.

2. Backend

4. Receive Expired Products:
   The backend receives the list of expired products from the database.
5. Send Prompt for Every Product:
   For each expired product, a query is sent to ChatGPT to retrieve updated ingredient data.

3. ChatGPT

6. Receive Ingredient Query:
   ChatGPT processes the query to retrieve the latest ingredient information for the product.

7. Return Ingredient Data:
   The updated ingredient information is sent back to the backend.

4. Backend (continued)

8. Update Ingredient Data:
   The backend updates the product's ingredient information with the data returned by ChatGPT.
9. Send Updated Product to ML Model:
   The backend sends the updated product information to the machine learning model for classification.

5. ML Model

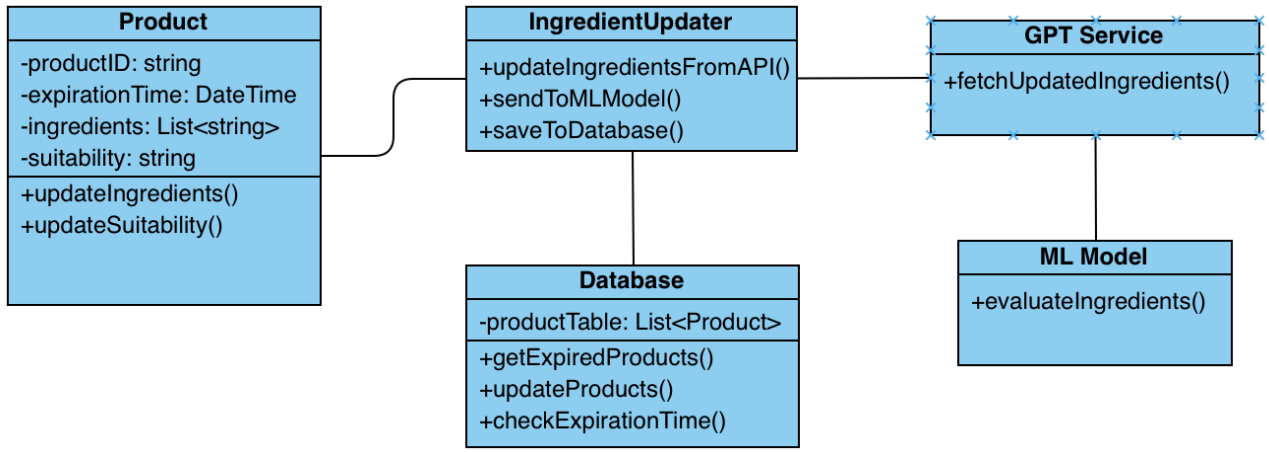
10. Receive Updated Product:
    The ML Model receives the updated ingredient data.
11. Evaluate and Label Updated Product:
    The model analyzes the ingredients and assigns a suitability label based on skin type compatibility.
12. Return Classification:
    The labeled product is sent back to the backend.

6. Backend (Final)

13. Save Updated Product:
    The backend saves the updated product information, including the new ingredient list and suitability labels, back to the database.

End State

The process ensures that expired product data is refreshed, reclassified, and stored in the database, maintaining the accuracy and reliability of the system's recommendations.



Class Diagram:

This Class Diagram represents the architecture and interactions within an ingredient update and suitability evaluation system for skincare products. The system ensures the product database remains accurate and relevant by periodically refreshing ingredient data using GPT APIs and re-evaluating product suitability with a machine learning model. It facilitates efficient updates to product information and suitability classifications.

1.Product

- **Attributes:**
  - productID: int – Unique identifier for the product.
  - expirationTime: Date – Expiration date of the product's ingredient data.
  - ingredients: List<String> – Current list of ingredients.
  - suitability: String – Suitability classification for specific skin types.
- **Methods:**

- updateIngredients() – Updates the product's ingredient list.
- updateSuitability() – Updates the suitability classification.

2.IngredientUpdater

- **Methods:**
  - updateIngredientsFromAPI() – Calls GPT API to retrieve updated ingredient data for a product.
  - sendToMLModel(productID: int, ingredients: List<String>): String – Sends the updated ingredient data to the ML model and retrieves suitability classification.
  - saveToDatabase(product: Product) – Saves the updated product details to the database.

3.GPTService

- **Methods:**
  - fetchUpdatedIngredients(): – Provides updated ingredient data for a product via GPT API.

4.MLModel

- **Methods:**
  - evaluateIngredients(): String – Evaluates the suitability of the ingredients for specific skin types and returns a suitability classification.

5.Database

- **Attributes:**
  - productTable: List<Product> – A list of all products in the database.
- **Methods:**
  - getExpiredProducts(): List<Product> – Fetches products with expired ingredient data.
  - updateProduct(product: Product) – Updates the product details in the database.
  - checkExpirationTime(): – Checks if the ingredient data has expired.

2.9. List All Products

This feature provides users with a comprehensive view of all skincare products available in the database. It also allows users to filter products by specific categories to quickly locate items of interest.

Purpose

The purpose of this function is to enhance user experience by:

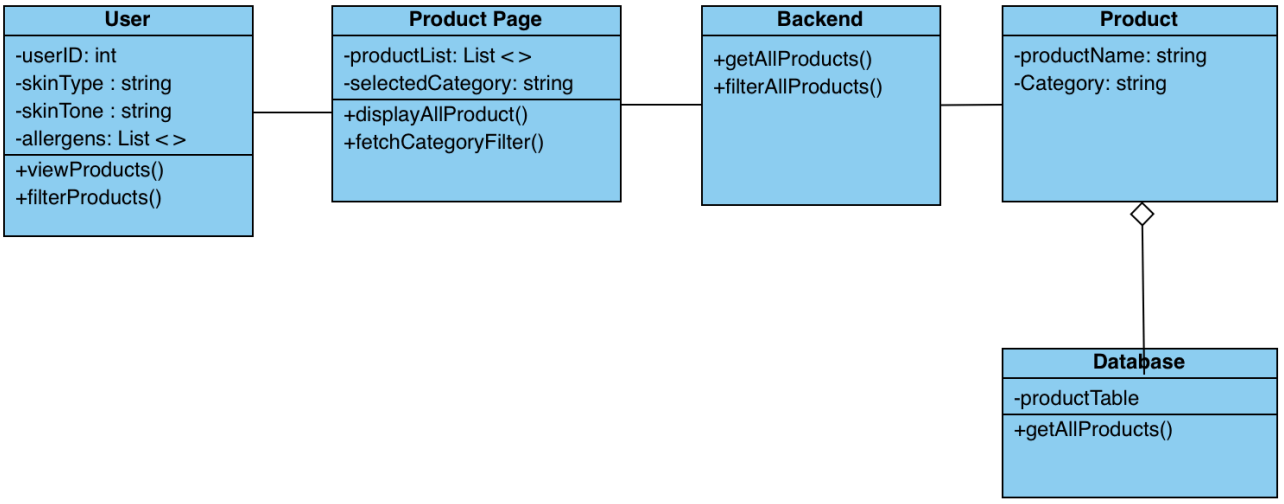
- Offering a centralized location for users to view all products.
- Allowing filtering by product categories for targeted searches.

Input

- **User Action**:
  - View all products: No input required; all products are displayed.
  - Apply a filter: The user selects a category (moisturizer, cleanser, toner, serum, sunscreen) from a dropdown or filter panel.

Output

- A list of all products or filtered products based on the selected category.
- Each product is displayed in a card format containing:
  - Product Name
  - Category



Class Diagram

This Class Diagram represents the architecture and interactions within a product display system that shows all available skincare products on a single page. The system enables users to browse and filter products by category, skin type, and other attributes.

Classes and Attributes:

1. User:

- **Attributes:**
  - userID: int – A unique identifier for the user.
  - skinType: string – The user's skin type.
  - skinTone: string – The user's skin tone, relevant for products like sunscreen.
  - allergens: List<> – A list of allergens that the user should avoid.
- **Methods:**
  - viewProducts() – Enables the user to view all products available in the system.
  - filterProducts() – Filters products based on user attributes like skin type, tone, or allergens.

2. Product Page:

- **Attributes:**
  - productList: List<> – A list of all products displayed on the page.
  - selectedCategory: string – The category selected by the user (e.g., cleanser, toner).
- **Methods:**
  - displayAllProduct() – Displays all products available, irrespective of category or skin type.
  - fetchCategoryFilter() – Fetch filters the products by category and updates the display.

3.Backend:

- **Methods:**
  - getAllProducts() - Get all products from database with GET method.
  - filterAllProducst() - Filter all products by category.

4. Product:

- **Attributes:**
  - productName: string – The name of the skincare product.

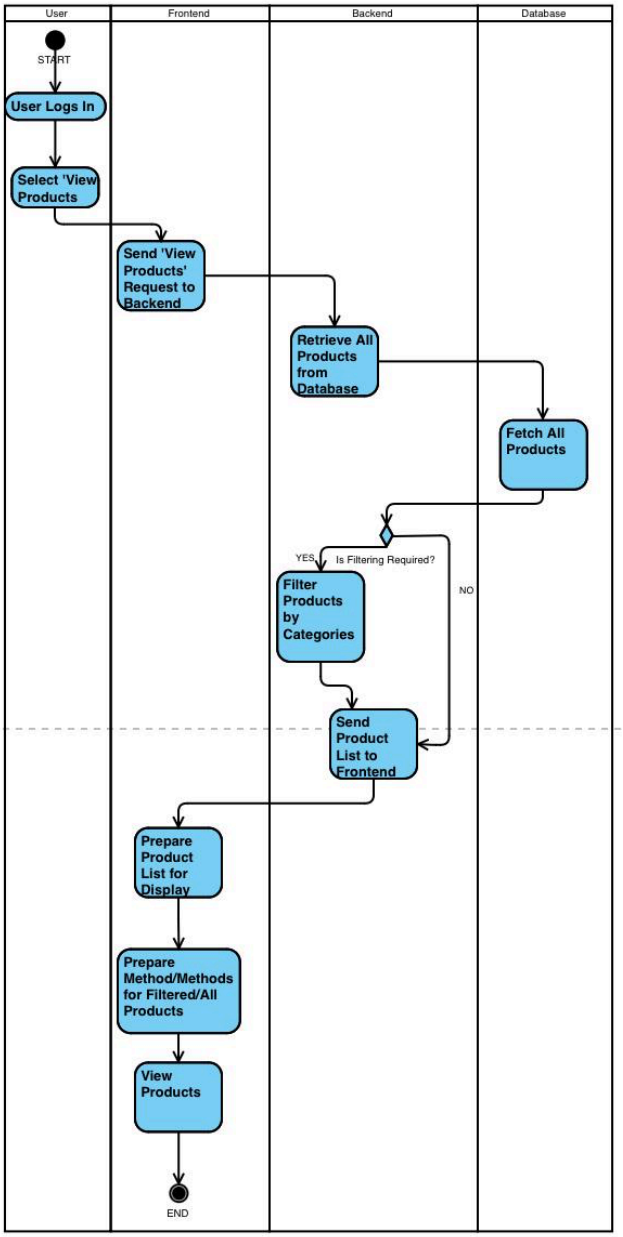
- Category: string – The category the product falls under.

5. Database:

- **Attributes:**
  - productTable – The main table storing product information.
- **Methods:**
  - getAllProducts() – Retrieves all products from the database and sends them to the product page for display.

Class Relationships:

- The **User** class interacts with the **Product Page** to view the list of products.
- The **Product Page** pulls product information from the **Database** and displays it to the user.
- The **Product** class holds information on individual items, including their name and categories.
- The **Backend** class get all products and filtered the products by category.
- The **Database** is responsible for storing all product data, which can be queried and displayed on the product page.



Activity Diagram

1. User

- **User Logs In**:
  - The process begins with the user logging into the system.
- **Select "View Products"**:
  - After logging in, the user chooses the "View Products" option to see the available products.

2. Frontend

- **Send "View Products" Request to Backend**:
  - The frontend sends a request to the backend to retrieve the list of products. This is typically achieved via an API call.

3. Backend

- **Retrieve All Products from Database**:
  - The backend receives the request and queries the database to fetch all product information.

4. Database

- **Fetch All Products**:
  - The database fetches all product entries and sends them to the backend for further processing.

5. Backend

- **Is Filtering Required?**:
  - The backend checks whether the user has applied any filters (e.g., category, price range, etc.).
  - **If Filtering is Required**:
    - The backend filters the product list according to the selected criteria.
  - **If No Filtering is Required**:
    - The backend skips this step and proceeds with the full list.
- **Send Product List to Frontend**:
  - The final list of products, whether filtered or unfiltered, is sent back to the frontend.

6. Frontend

- **Prepare Product List for Display**:
  - The frontend prepares the received product list for user display. This includes formatting and arranging the products visually.
- **Prepare Methods for Filtered/All Products**:
  - The frontend ensures that filtering options are functional, allowing users to interact with the displayed list as needed.

7. User

- **View Products**:
  - Finally, the user views the list of products displayed on the frontend. They can further refine the list using filters if desired.

3. Non-Functional Requirements

3.1. Development Environment

Machine Learning

- **Languages:** Python
- **IDE:** PyCharm
- **Algorithms:** Random Forest (Softmax), SVM, SHAP

Frontend

- **Languages**: HTML, CSS, JavaScript,TypeScript
- **Frameworks/Libraries**:
    - React (Node.js for SSR)
    - Angular
    - Bootstrap

Backend

Backend will be written in Python. OpenAI's API is accessible via HTTPS requests, so we can use any programming language that supports making HTTP requests, including Python.

- **Languages**: Python
- **Frameworks**:
    - Flask
- **APIs**: REST, ChatGPT

Database

PostgreSQL will be the sole database used to manage structured data, including user profiles, product information, and skin type test results. Its robust architecture, scalability, and support for complex queries ensure data consistency, integrity, and high performance.

- **Database Management System (DBMS):** PostgreSQL

Backup Strategy:

- Weekly full database backups with daily incremental backups to ensure data security and recovery.

DevOps and Hosting

- GitHub

3.2. Security

- SSL for secure data transfer.
- Passwords stored with hashing (bcrypt).
- API keys stored securely (environment variables).
- Role-based access control (RBAC) to restrict sensitive data access.
- Data encryption at rest and during transmission to safeguard user information.

3.3. Scalability

Scalable database and API architecture to handle increased user data.

3.4. Testing

Unit, integration, and user acceptance tests for all features.

3.5. Data Privacy

Ensure user data is encrypted and stored securely.

4. References

1. Kumar, K., Sinha, V., Sharma, A., Monicashree, M., Vandana, M.L., Vijay Krishna, B.S.: AIassisted college recommendation system. In: Intelligent Sustainable Systems: Proceedings of ICISS 2022, pp. 141–150. Springer Nature Singapore, Singapore (2022) DOI:10.1007/978-981-99-9018-4_28

2. Saidah, S., Fuadah, Y.N., Alia, F., Ibrahim, N., Magdalena, R., Rizal, S.: Facial skin type classification based on microscopic images using convolutional neural network (CNN). In: Proceedings of the 1st International Conference on Electronics, Biomedical Engineering, and Health Informatics: ICEBEHI 2020, 8–9 Oct, Surabaya, Indonesia, pp. 75–83. Springer Singapore (2021) DOI: 10.1007/978-981-33-6926-9_7
3. Vinutha, M., Dayananda, R.B., Kamath, A.: Personalized skincare product recommendation system using content-based machine learning. In: 2024 4th International Conference on Intelligent Technologies (CONIT), pp. 1–9. IEEE, Bangalore, India (2024). DOI: 10.1109/CONIT61985.2024.10626458
4. Jadhav, S., Memane, D., Supekar, K., Shinde, S., & Jadhav, T. (2021). *A Personalized Skincare Recommendation System Using Machine Learning*. *CIENCIENGN*, 11(1). DOI: 10.52783/cienceng.v11i1.127
5. Lokesh, B., Devarakonda, A., Srinivas, G., & Naik, N. K. (2024). *Intelligent Facial Skin Care Recommendation System*. *AFJBS*, 6(Si2), 1822-1830. DOI: 10.33472/AFJBS.6.Si2.2024.1822-1830

APPENDIX B: DESIGN SPECIFICATIONS DOCUMENT

COMP4910 Senior Design Project 1, Fall 2024
Advisor: Assoc. Prof. Dr. Ömer ÇETİN

GLOWG: Personalized Skincare Powered by AI
High Level Design
Design Specifications Document

Revision 1.0
20.01.2025

By:
Ceren Sude Yetim 21070001045
İrem Demir, 20070001029
Gizem Tanış, 20070001047
Ece Topuz, 21070001057

Revision History

Revision	Date	Explanation
1.0	20.01.2025	Initial high level design

Table of Contents

1. Introduction

Purpose:

The purpose of this project is to develop the **GlowGenie Application** based on the foundation laid in the **GlowGenie Requirements Specification Document (RSD)**. GlowGenie is designed to provide a user-friendly, AI-powered skincare solution to address challenges in determining skin types, evaluating product suitability, and providing personalized recommendations. This Detailed Software Design (DSD) document outlines the design and implementation details of the GlowGenie application, utilizing **Python for backend, React.js for frontend**, and **PostgreSQL** for database management.

Main Functions of GlowGenie System:

1. **User Account Operations:**
    - Login, registration, and logout functionalities.
    - Ability to update user profiles, including skin type, skin tone, and allergens.
    - Password reset functionality with secure email-based verification.
2. **Skin Type Test:**
    - An AI-based interactive questionnaire to determine the user's skin type (Dry, Oily, Combination, Normal).
    - Results are saved in the user's profile for future use.
3. **Product Suitability Feedback:**
    - Analyze skincare products to determine their compatibility with the user's skin type, tone, and allergens.
    - Provide ingredient-based feedback with visual indicators (green/red framing) for suitability.
4. **Generate Product Recommendations:**
    - Suggest personalized products tailored to the user's preferences and skin profile.
    - Allow filtering by product categories like cleansers, moisturizers, sunscreens, toners, and serums.
5. **Update Product Ingredients:**
    - Periodically update product ingredient data via GPT integration.
    - Re-evaluate product suitability based on updated information.
6. **List All Products:**
    - Display a comprehensive list of products with filtering options by category.
    - Provide detailed information for each product, including ingredients and suitability feedback.

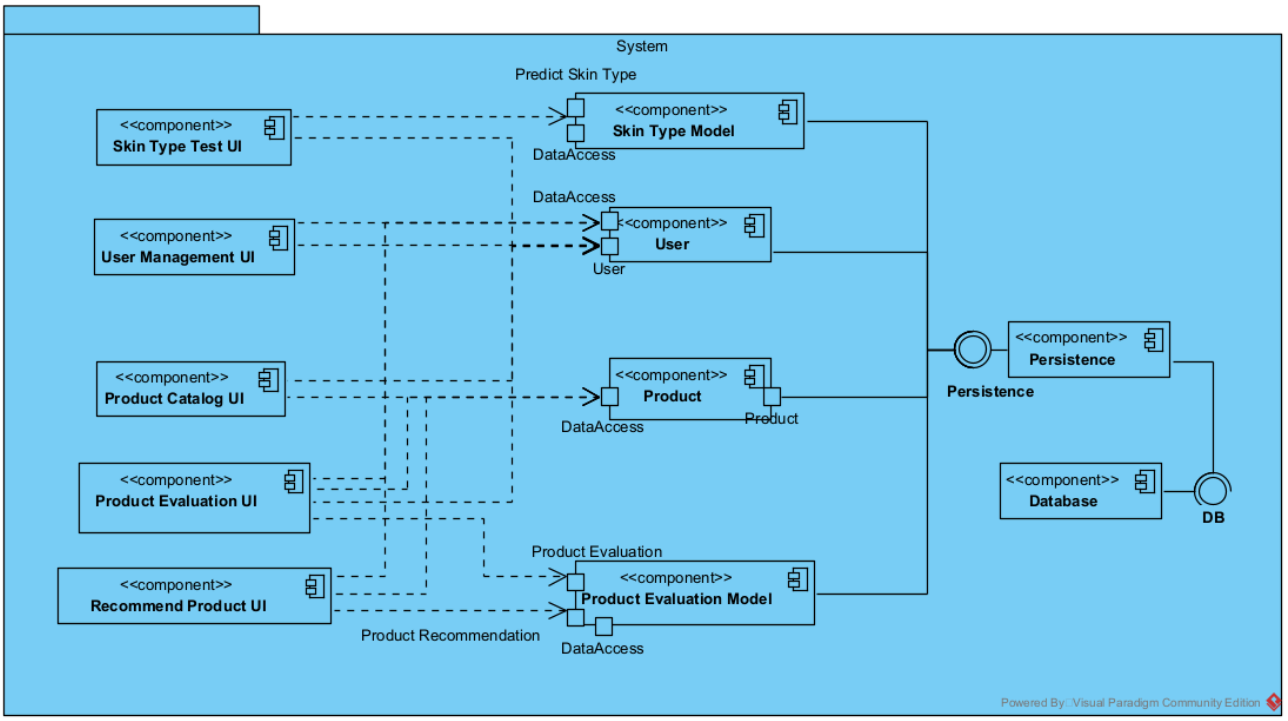
Design Basis and Methodology

The design of GlowGenie is based on the **GlowGenie Requirements Specification Document (RSD), Revision 2.0**, in the file `GlowGenie-RSD-2025-01-12.doc`[1]. The design process used to produce this document conforms to the organizational specifications provided in [2]. This DSD document serves as a continuation of the RSD, detailing the system's architectural choices, structural breakdown, and implementation specifics. The notation used to describe the design of the GlowGenie Application is primarily **UML**, which adheres to the organizational standards outlined in [3].

The design methodology adheres to **ISO/IEC 9001 software quality standards**, ensuring a structured and reliable process for developing the GlowGenie application. This standard emphasizes **usability**, **performance efficiency**, and **security**, aligning with internationally recognized best practices for quality management systems. To ensure clarity and consistency, **Unified Modeling Language (UML)** is extensively used throughout the document, detailing the architecture, component interactions, and class designs. This approach guarantees that the system meets user requirements while maintaining high quality and scalability.

2. GLOWG System Design

The GLOWG system design focuses entirely on the **software subsystem**, as no hardware components are required for the development and implementation of this project. This section provides an overview of the system, identifying its components and their interactions, and includes a UML Component Diagram to visualize the relationships among the components including **Skin Type Test UI**, **User Management UI, Product Catalog UI, Recommend Product UI, User, Product, Product Evaluation Model**, **Skin Type Model, Persistence** and **Database**. These components work together to provide a personalized and user-friendly experience for evaluating skincare products. The design decisions ensure scalability, maintainability, and extensibility, making the system adaptable to future requirements.



3. GLOWG Software Subsystem Design

Since the GLOWG project is a software-only system, this section focuses on the design of the **software subsystem**, which encompasses the entire system. Below is the detailed explanation of the architectural style chosen for the GLOWG system, along with a justification of the design decisions.

3.1. GLOWG Software System Architecture

The GlowGenie system employs a **Layered Architecture**, a widely used architectural style in software engineering. It structures the application into three primary layers, ensuring modularity, scalability, and separation of concerns.

Architecture Layers:

1. **Presentation Layer**:
    - **Role**: Responsible for the user interface and interaction.
    - **Technology**:
    - Developed using React.js (with Node.js for Server-Side Rendering) and Angular for building dynamic and responsive user interfaces. Styled with Bootstrap for consistent and responsive design.
    - **Features**:
        - Handles user interactions (e.g., login, product filtering, and test submissions).

        - Displays data from the backend using RESTful APIs.
2. **Application Layer**:
    - **Role**: Serves as the bridge between the presentation and data layers.
    - **Technology**: Implemented in Python, using Flask, RESTful API and GPT API.
    - **Features**:
        - Processes user inputs and applies business logic.
        - Implements AI-powered features, such as skin type analysis, compatibility feedback, and product ingredient evaluation via GPT API.
        - Facilitates secure and seamless communication between the frontend, database, and external APIs.
3. **Data Layer**:
    - **Role**: Manages persistent data storage and retrieval.
    - **Technology**: Uses **PostgreSQL** for relational database management.
    - **Features**:
        - Stores user profiles, product information, test results, and recommendation data.
        - Ensures data integrity and security through proper indexing, constraints, and role-based access.

Justification for Layered Architecture:

- **Scalability**: Each layer can be scaled independently to handle increased traffic or data.
- **Maintainability**: The separation of concerns allows developers to modify one layer without affecting others.
- **Security**: Sensitive data is handled securely at the application and data layers, reducing exposure risks.
- **Flexibility**: New features can be added or existing features updated within their respective layers.

Design Decisions and Justification

The design decisions for the GLOWG system are guided by the need for scalability, modularity, and user-centric design. Below is a detailed justification for the selected architectural style:

1. **Layered Architecture:**
    - **Why Chosen:** Layered architecture ensures clear separation of responsibilities, making the system easier to extend, debug, and maintain. It also allows independent development of each layer by different teams.
    - **Benefits:** Facilitates the integration of new features (e.g., additional ML models or external APIs) without impacting other layers.
2. **Integration of Machine Learning Models:**
    - **Why Chosen:** Machine learning provides the ability to classify and evaluate product suitability dynamically based on user attributes and product data. It enhances the system's intelligence and adaptability.
    - **Benefits:** Provides transparent and explainable feedback to users, ensuring trust and personalization.
3. **Use of External APIs:**
    - **Why Chosen:** Leveraging external APIs enables the system to access up-to-date product data and enhance analysis without duplicating existing data sources.
    - **Benefits:** Reduces the need for extensive data entry and maintenance, while ensuring accuracy and comprehensiveness.
4. **Scalable Database Design:**
    - **Why Chosen:** PostgreSQL was selected for its support of complex queries and scalability. It handles the relational data of users, products, and feedback effectively.
    - **Benefits:** Ensures efficient data storage and retrieval, supporting future growth in the number of users and products.
5. **Frontend Technology (React.js):**
    - **Why Chosen:** React.js was selected for its ability to build highly interactive user interfaces with reusable components.

    - **Benefits:** Provides a responsive and dynamic user experience, which is essential for attracting and retaining users.

3.2. GLOWG Software System Structure

The system structure of GlowGenie is organized into several primary components, each fulfilling specific roles within a layered architecture. These components have been designed with a focus on modularity, scalability, and maintainability, ensuring that the system is adaptable to future enhancements. The core components include the **Frontend**, **Backend**, **Machine Learning Models**, and **Database** packages, with the integration of external APIs handled within the backend.

Key Components and Their Roles:

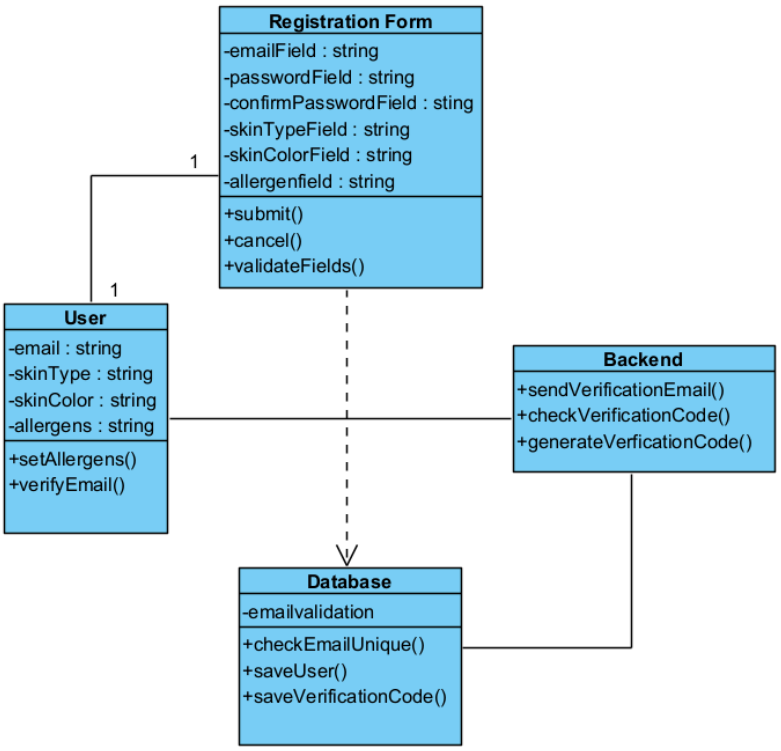
Frontend Package:

- **Responsibilities:** Handles user interactions, including skin type tests, product browsing, and feedback display.
- **Features:**
    - User-friendly interface for personalized skincare recommendations.
    - Real-time filtering of products based on categories, skin types, and preferences.
    - Visual cues for product suitability (e.g., green/red framing).
- **Sub-components:**
    - **User Management UI:** Manages registration, login, profile updates, and skin type test interactions.
    - **Product Catalog UI:** Displays products, allows filtering, and shows detailed product information.

Backend Package:

- **Responsibilities:** Manages core business logic, API services, and interactions with the database and external APIs.
- **Features:**
    - Secure user authentication and profile management.
    - Product data management, including ingredient-based filtering.
    - Integration with external APIs for ingredient data (e.g., GPT APIs).
- **Sub-components:**
    - **User Service:** Implements logic for user registration, authentication, and profile management.
    - **Product Service:** Handles product categorization, filtering, and compatibility analysis.
    - **GPT API Integration:** Manages communication with external APIs to fetch updated ingredient data.
    - **Logging & Security:** Ensures secure operations and logs key activities for debugging and monitoring.

Registration Class Diagram:

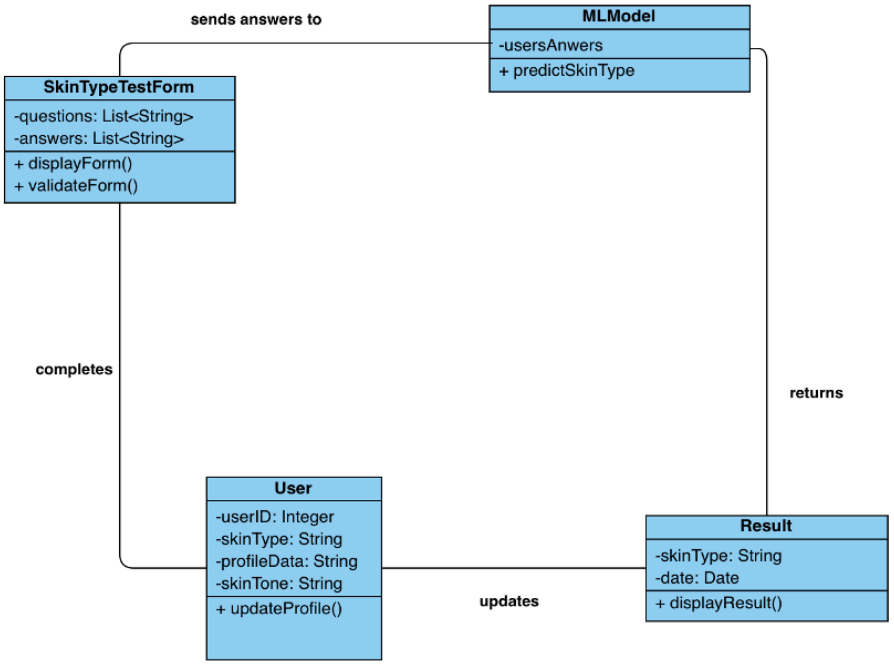


Machine Learning Models Package
Responsibilities:
Manages product compatibility evaluations and updates, ensuring accurate skin type classification and suitability labeling for skincare products based on AI models.

Features:

- **Skin Type Prediction:** Determines users' skin types through a questionnaire.

Skin Type Prediction Class Diagram:



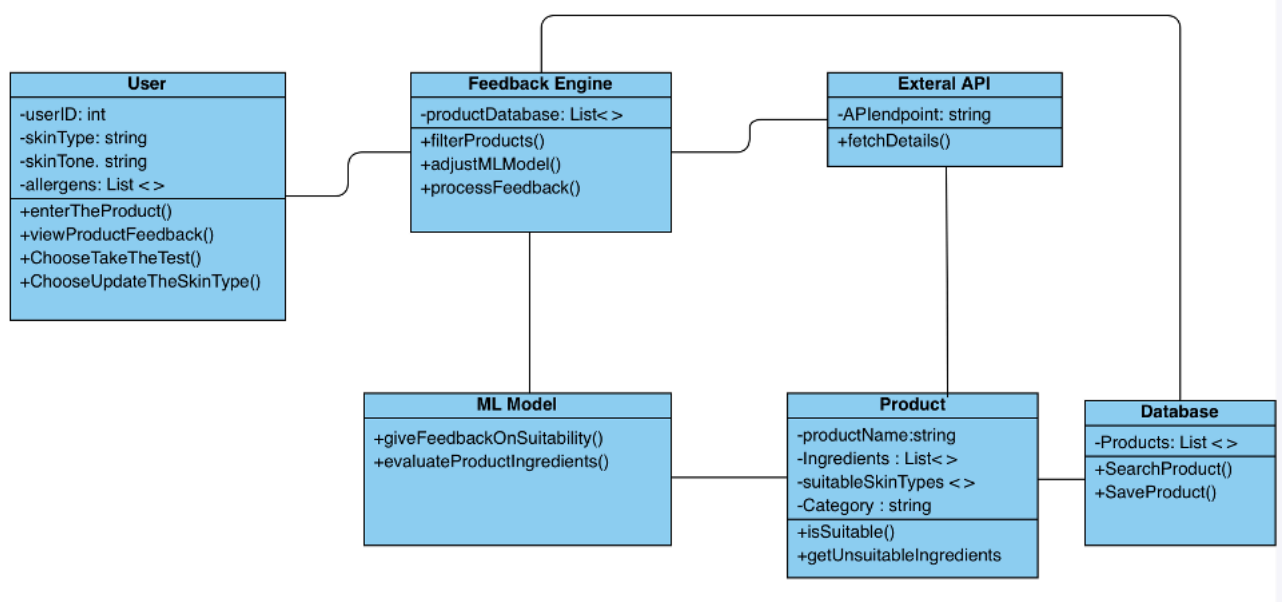
This Class Diagram illustrates the core components and relationships within a skin type detection system. The diagram represents the flow where a user completes a form, submits answers, and the machine learning model processes these answers to predict the skin type, updating the user's profile with the results.

Classes and Attributes:

1. **SkinTypeTestForm:**
    - **Attributes:**
        - questions: List<String> – A list of questions presented to the user.
        - answers: List<String> – A list of responses provided by the user.
    - **Methods:**
        - displayForm() – Displays the test form to the user.
        - validateForm() – Validates user responses for completeness and accuracy.
2. **MLModel:**
    - **Attributes:**
        - usersAnswers – Answers submitted by the user.
    - **Methods:**
        - predictSkinType() – Analyzes responses and predicts the user's skin type.
3. **Result:**
    - **Attributes:**
        - skinType: String – The detected skin type.
        - date: Date – The date the test was conducted.
    - **Methods:**
        - displayResult() – Displays the test results to the user.
4. **User:**
    - **Attributes:**
        - userID: Integer – Unique identifier for the user.
        - skinType: String – The user's skin type.
        - profileData: String – Additional profile information.
        - skinTone: String – The user's skin tone.
    - **Methods:**
        - updateProfile() – Updates the user's profile based on test results.

- **Product Suitability Evaluation:** Classifies products based on ingredient data and predicts their compatibility with various skin types.
- **Ingredient Analysis Feedback:** Identifies specific ingredients contributing to a product's suitability or unsuitability for a given skin type.

Ingredient Analysis Feedback Class Diagram:



This Class Diagram illustrates the architecture and interactions within the product suitability feedback system. The system evaluates skincare products based on user attributes (skin type, skin tone(for sunscreen) and allergens) and provides feedback using machine learning models and external product databases.

Classes and Attributes:

1. **User:**
   - **Attributes:**
     - userID: int – Unique identifier for the user.
     - skinType: string – User's skin type (e.g., oily, dry, sensitive).
     - skinTone: string – User's skin tone (for evaluating products like sunscreen).
     - allergens: List<> – List of allergens to avoid.
   - **Methods:**
     - enterTheProduct() – Allows the user to input a product name.
     - viewProductFeedback() – Displays the feedback on product compatibility.
     - ChooseTakeTheTest() – Directs the user to take a skin type test.
     - ChooseUpdateTheSkinType() – Updates user's skin type profile.
2. **Feedback Engine:**
   - **Attributes:**
     - productDatabase: List<> – A list of products available for feedback and analysis.
   - **Methods:**
     - filterProducts() – Filters products based on user attributes and queries.
     - adjustMLModel() – Adjusts machine learning parameters for better predictions.
     - processFeedback() – Processes and refines product suitability feedback.
3. **ML Model:**
   - **Methods:**
     - giveFeedbackOnSuitability() – Generates feedback on whether a product is suitable for the user.
     - evaluateProductIngredients() – Evaluates the ingredients of a product to determine its compatibility.

4. **Product:**
   - **Attributes:**
     - productName: string – The name of the skincare product.

        - Ingredients: List<> – A list of ingredients present in the product.
        - suitableSkinTypes<> – Skin types for which the product is suitable.
        - Category: string – Product category (e.g., moisturizer, sunscreen).
    - **Methods:**
        - isSuitable() – Determines if the product is suitable for the user's skin type.
        - getUnsuitableIngredients() – Lists ingredients that may cause issues for the user.
5. **External API:**
    - **Attributes:**
        - APIendpoint: string – The endpoint used to fetch external product details.
    - **Methods:**
        - fetchDetails() – Fetches product data from external databases.
6. **Database:**
    - **Attributes:**
        - Products: List<> – A list of stored products.
    - **Methods:**
        - SearchProduct() – Searches for products in the database.
        - SaveProduct() – Saves new product entries.

- **Retraining and Updating Models:** Keeps models up-to-date by incorporating new user data, product information, and feedback.

Sub-components:

1. **Skin Type Test:**
    - AI-based questionnaire designed to predict users' skin types using a pre-trained model.
2. **AI Model Integration:**
    - Implements the logic for integrating machine learning models to evaluate and label products for skin type suitability.
3. **Model Retraining Service:**
    - Handles retraining of the skin type prediction and product suitability models, incorporating updated ingredient and user feedback data to improve accuracy.
4. **Recommendation Engine:**
    - Generates personalized skincare recommendations based on the compatibility between user skin types and labeled product data.
    - Provides ingredient-based feedback to users for transparency and informed decision-making.

Skin Type Prediction Model

1. **Input Collection:**
    - The user completes a skin type test, where each question represents a feature in the model (e.g., frequency of dryness, oiliness, sensitivity, etc.).
    - At first, all questions will directly contribute as features, ensuring a straightforward and comprehensive representation of the user's skin characteristics.
2. **Model Processing:**
    - The responses are processed and fed into the pre-trained skin type prediction model.
    - The model is trained to classify skin types (e.g., dry, oily, combination, sensitive) based on labeled training data.
3. **Prediction:**
    - The model outputs a single skin type as the prediction for the user.
    - The result is saved in the user’s profile and used for product recommendations.

Evaluating and Labeling Products for Skin Types Model

1. **Product Data Preparation:**

    - Ingredient data for each product is retrieved from the database.
    - This data serves as input to the model for suitability evaluation.
    - No additional conversion into feature vectors is considered as Random Forest can directly handle structured input.
2. **Model Processing:**
    - A Random Forest model is used to predict product suitability.
    - The model evaluates how suitable a product is for each skin type (e.g., 85% suitable for dry skin, 20% for oily skin, etc.).
    - Softmax is applied to the output to provide a percentage for each skin type.
3. **Labeling:**
    - The product is labeled for each skin type as either **suitable** or **unsuitable** based on the highest suitability percentage or a threshold.
    - Using SHAP (SHapley Additive exPlanations), the model identifies which ingredients contributed to the suitability or unsuitability of the product for each skin type.
    - The reasons for labeling (i.e., the specific ingredients responsible for suitability or unsuitability) are saved to the database alongside the product.
4. **Database Update:**
    - The product data is updated with its suitability percentages, suitability labels for each skin type, and SHAP analysis results (ingredient-level explanations).

Database Package:

- **Responsibilities:** Manages user data, product information, and feedback in a structured database.
- **Features:**
    - Persistent storage of user profiles, products, and recommendations.
    - Efficient data retrieval and updates.
- **Sub-components:**
    - **User Repository:** Handles CRUD operations for user-related data.
    - **Product Repository:** Manages product-related data in the database.
    - **Recommendation Repository:** Stores and retrieves generated recommendations and feedback.

3.3. GLOWG Software System Environment

The GLOWG software subsystem is designed to operate in a robust and scalable environment, leveraging modern hardware, system software, and middleware to ensure optimal performance and reliability. This section provides a detailed description of the target environment, programming tools, and supporting software.

3.3.1 System Software Environment

The system software environment comprises the backend, frontend, database, and middleware technologies that collectively run the GLOWG application.

1. **Backend**:
    - **Programming Language**: Python 3.10.
    - **Frameworks**:
        - Flask: Lightweight web framework for RESTful API implementation.
    - **APIs**: Integration with GPT APIs for ingredient data updates and external AI enhancements.

    Machine Learning Integration:

    - **Scikit-learn:** For implementing the Random Forest model used for product suitability evaluation.
    - **Numpy:** For numerical computations and efficient matrix operations.

    - **Pandas:** For data manipulation and preprocessing, particularly for handling datasets of ingredients and skin type responses.
    - **Softmax (via Scikit-learn/Numpy):** To compute suitability percentages for products across different skin types.
    - **SHAP (SHapley Additive exPlanations):** For generating interpretability insights for product suitability and storing ingredient-based explanations in the database.
2. **Frontend**:
    - **Programming Language**: JavaScript/TypeScript, HTML, CSS
    - **Frameworks**:
        - React.js 18: For building responsive and dynamic user interfaces.
        - Node.js: For server-side rendering and improved performance.
    - **Styling**: Bootstrap 5 for consistent and responsive UI design.
3. **Database**:
    - **Database Management System**: PostgreSQL 15.
    - **Features**:
        - Support for advanced queries and indexing for faster data retrieval.

3.3.2 Development Tools

The development environment for the GLOWG application uses the following tools and platforms:

1. **Integrated Development Environment (IDE)**:
    - Visual Studio Code for coding, debugging, and integrating frontend and backend services.
2. **Version Control**:
    - Git for source code management and collaborative development.
    - GitHub as the central repository for version control and issue tracking.
3. **Testing Tools**:
    - **Postman**: For API testing and verification.
    - **Selenium**: For automated testing of the user interface.
    - **Pytest**: For unit and integration testing in the backend.

3.3.3 Justification for Environment Choices

1. **Modern Frameworks and Tools**:
    - React.js and Flask ensure efficient development and user-friendly interfaces.
    - PostgreSQL provides robust data management capabilities.
2. **Middleware**:

- **Authentication**:
    - **JWT** (jsonwebtoken)
- **Body Parsing**:
    - **express.json()** and **express.urlencoded()** (Built-in in Express)
- **Input Validation**:
    - **express-validator**
- **Error Handling**:
    - Custom middleware in Express
- **Caching**:
    - **cache-manager** (in-memory, Redis, etc.)

4. GLOWG Software System Detailed Design:

Detailed design will be carried out in the context of the COMP 4920 course next semester.

5. Testing Design

We will use **unit testing**, **integration testing**, and **user acceptance testing (UAT)** to ensure the application is reliable and user-friendly. Unit testing will validate individual components, integration testing will ensure

seamless interactions between modules, and UAT will confirm the application meets user expectations. Together, these methods ensure functionality, reliability, and user satisfaction.

References

1. GlowGenie-RSD-2025-01-12.doc (Requirements Specification Document).

APPENDIX C: PROJECT MANAGEMENT DOCUMENTS

APPENDIX C1: PROJECT PLAN

COMP 4910 GLOWG: GlowGenie, Project Plan, 10.11.2024, v1.0																	
Task No	Task Name	Weeks															Any Additional Notes
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
1	Project Planning & Requirements Gathering	X	X	X	X										X		
2	Literature Review					X	X	X							X		
3	ChatGPT Integration Research					X					X				X		
4	Data Collection Requirements										X			X	X		
5	Machine Learning Algorithms Research							X		X	X	X	X	X	X	X	
6	UI Design													X	X	X	
7	Database Structure & Design Planning												X	X	X		
8	Backend Frameworks & API Design Study							X	X	X	X				X		
9	Presentation Preparation															X	
Form: GLOWG-Project-Plan-2024-11-10-v1.0.xlsx																	

APPENDIX C2: PROJECT EFFORT LOG- CONSOLIDATED

COMP 4910/4920 Project Effort Log, Project Code: GLOWG, 10.11.2024, v1.0										
Week	Dates	İrem Demir		Ceren Sude Yetim		Ece Topuz		Gizem Tanış		Total Weekly Effort in Man-Hours
		Work Done	Total Hours Spent	Work Done	Total Hours Spent	Work Done	Total Hours Spent	Work Done	Total Hours Spent	
Week 1	30.09.24 - 04.10.24	Define project problem, objectives, and scope. Creation of the team.	4	Formed a team for the project and defined some problems to decide on the topic of the project.	4	A suitable team was formed for the project and organization was ensured. Project Problem defined.	4	Contributed to team formation by defining project goals and discussing potential challenges with team members to clarify the project direction.	4	16,00
Week 2	07.10.24 - 11.10.24	Project selection. Clearly define the project scope. Advisor selection has been completed.	4	Consulted some of the advisor teachers for the project and chose leads for main areas of the project. Selected the project topic.	4	Defining a project scope based on a detected problem and selecting the project topic accordingly.Advisor teacher has been determined	4	Participated in the project topic selection by discussing identified problems and clarifying the project scope. Assisted in consulting with advisor teachers and selecting team leads for key areas.	4	16,00
Week 3	14.10.24 - 18.10.24	Determined the project's goals and objectives, and created a preliminary draft of the project proposal form.	5	Decided on goals of the project and it's objectives and prepared a draft project proposal form.	5	Project goals and objectives were determined and a draft project proposal was prepared.	5	Contributed to defining the project’s goals and objectives, assisting in the creation of a draft proposal form to outline the initial project	5	20,00

								framework.		
Week 4	21.10.24 - 25.10.24	Project goals determined and prepared the PAF documentation.	5	Worked on the draft project proposal form and readied the final version of PAF.	5	The project proposal draft was detailed and the PAF file was completed and delivered.	5	Assisted in finalizing project goals and contributed to refining the draft proposal. Helped prepare and complete the PAF documentation for submission.	5	20,00
Week 5	28.10.24 - 01.11.24	Investigate API documentation, integration requirements, and limitations for ChatGPT in a skincare context.	6	Conducted research on integrating ChatGPT into the project by examining similar projects to gain insights into successful implementation strategies.	6	Similar projects in the relevant field were examined and information that could be a reference for the project was collected.	6	Researched API integration options and potential limitations for implementing ChatGPT in a skincare context. Contributed insights from related projects to support integration planning.	6	24,00
Week 6	04.11.24 - 08.11.24	Research academic sources on skincare science, AI in health, and personalized recommendations.	6	Outlined key tasks for the design process and conducted research on potential technologies those most suitable for the project. Continued examining similar projects to gain further insights.	6	A literature review on AI in health and AI-driven personalized recommendations was conducted from academic sources.	6	Conducted a literature review on skincare science and the use of AI for personalized recommendations. Assisted in outlining design tasks and explored potential technologies to support project development.	6	24,00

Week 7	11.11.24-15.11.24	Analysis of similar projects: Review existing skincare recommendation systems and identify gaps to address.	6	Conducted research on technologies utilized in similar projects to identify and evaluate the most suitable options for implementation in our project.	5	By analyzing skin care recommendation systems, the existence of similar applications and their place in the literature were examined. Additionally, the role and contributions of artificial intelligence in such systems were investigated.	6	Developed a preliminary framework for a skincare recommendation system, focusing on defining the user interface and core functionalities based on the insights from similar applications and literature reviews.	6	23,00
Week 8	18.11.24-22.11.24	Midterm exams week	0	Midterm exams week	0	Midterm exams week	0	Midterm exams week	0	0,00
Week 9	25.11.24-29.11.24	Identified shortcomings in our current approach and generated ideas for innovative features to improve the system. Assessed the practicality of these concepts by examining their technical requirements.	7	Planning the interactions between the components of the project and evaluating their compatibility with one another.	7	Organizing how the project's components will interact and assessing how well they work together.	7	Evaluated the feasibility of proposed features and contributed to aligning project components with overall objectives.	7	28,00

Week 10	02.12.24-06.12.24	Researched requirements for GPT API integration, including backend frameworks, libraries, and programming languages for compatibility with machine learning workflows.	8	Designing the structure of the machine learning model, including determining its architecture, feature selection, and overall approach to ensure alignment with the project's objectives and requirements	8	Studies were conducted on the solutions we obtained in the literature reviews, including determining its architecture, feature selection, and overall approach to ensure that it was aligned with the project's goals and requirements.	8	Researched database optimization techniques and security measures to ensure data integrity and protection. Contributed to identifying potential challenges and solutions for smooth implementation.	8	32,00
Week 11	09.12.24 - 13.12.24	Researching the backend structure for the Glow Genie project. My focus was on understanding the necessary technologies, tools, and design patterns to create a robust and scalable backend system	9	Planned the project's functionality and user interaction flow, ensuring seamless integration of components and proposing interface ideas to enhance usability.	9	Designed the functionality and user interaction flow of the project, emphasizing the smooth integration of components and putting forward creative interface concepts to enhance usability.	8	Focused on researching and designing the database structure to ensure efficient data management for the Glow Genie project.	8	34,00

Week 12	16.12.24 - 20.12.24	Researching UML diagrams to determine which ones would be most suitable for the project. I started designing UML diagrams, including use case, class, and activity diagrams. Additionally, as a team, we roughly sketched the project interface to visualize the overall structure.	7	Researched suitable machine learning algorithms, defined their integration strategy, and developed UML diagrams to represent the system's architecture and processes.	7	Conducted research on appropriate machine learning algorithms and started drafting UML diagrams to represent system structure and processes.	7	Focused on researching database optimization techniques and began creating UML diagrams, including class and entity-relationship diagrams, to design the data architecture for the project.	7	28,00
Week 13	23.12.24 - 27.12.24	Working on the UML diagrams, refining the designs to ensure they accurately represent the system's structure and functionality. Conducted research to determine what components and features should be included in the project interface. Planning a demo version of the interface, outlining the key elements and their placement	11	Based on further research, I refined the UML diagrams and requirements specification document and finalized the machine learning algorithm for the project.	10	Refined the UML diagrams and requirements specification based on further research, finalizing the system's entry and registration mechanism.	9	Refined the database design and UML diagrams based on further research, focusing on optimizing the data architecture for the project.	9	39,00

Week 14	30.12.24 - 03.01.25	This week, due to major changes in the our project, I revised all our forms and UML diagrams to reflect the new direction and updated functionality. I redefined key functions, adjusted the scope to align with the project's goals, and contributed to redesigning how the system components interact. I researched and integrated updates to the backend technologies to improve system performance and ensure seamless operation with the revised requirements. I also worked on updating user interface designs for improved usability and created detailed UML diagrams to clarify system workflows, ensuring the project remains cohesive and on track.	22	The project underwent significant changes, so I revised the Requirement Specification Document (RSD) to align with the updated direction. I introduced new functions for the project, finalized how the machine learning models will operate, and defined the structure of the dataset. Additionally, I researched skin type characteristics to design accurate skin type test questions and answer choices, helped editing the survey we made to gather data. I also created UML diagrams for some of the functions, designed interfaces for	32	Significant changes were made within the scope of the project and RSD was reorganized to adapt to the updated goals. In this process, new functions were defined for the project and these functions were explained in detail. Research was conducted to design the right skin type test questions and answer options for the machine learning model to be developed. A survey was prepared and organized for data collection. In addition, UML diagrams were created for specific functions and detailed explanations were added for each diagram. A demo was designed for the project's user interface and included in the development	27	This week, significant changes were made within the scope of the project, and the Requirement Specification Document (RSD) was reorganized to adapt to the updated goals. I conducted research to design accurate skin type test questions and answer options for the machine learning model. A survey was prepared and organized for data collection. I created UML diagrams for some of the project's functions and added detailed explanations for each diagram. Additionally, I contributed to the design of the user interface and participated in the demo development process. I reviewed and redesigned the database table structures, implementing	22	103,00

				some features, and worked on improving the overall project flow.		process.		necessary improvements to optimize data flow. I also reviewed the overall system flow to improve the project's progress and ensure smooth operation.		
Week 15	06.01.25 - 10.01.25	Some missing functions have been added to the RSD report. Focused on completing the missing UML diagrams for project, ensuring that all necessary diagrams were finalized. Also prepared for the DSD report by organizing the required content and began drafting its initial sections. Additionally, worked on creating packages for the backend structure to enhance organization.	13	Existing functions were reviewed and refined, and a new function was added. UML diagrams were created to represent the refined and newly implemented functions, providing a clearer understandin g of their structure and interactions. Additionally, significant progress was made on the Data Structure Design (DSD).	11	The RSD report was meticulously reviewed, missing functions and UML diagrams were completed and added to the report. In addition, significant progress was made within the scope of the DSD report, and detailed studies were carried out in the analysis and design stages. In this direction, the reports were made more comprehensive and understandable during the project development process.	13	This week, I finalized the UML diagrams for the Requirements Specification Document (RSD) and made adjustments. We then transitioned to the Design Specification Document (DSD) and added new content. I also researched database design, focusing on table relationships, optimization techniques, security, data integrity, and backup strategies.	12	49,00

Week 17	13.01.2025 - 17.01.25	Finalized the final report, DSD report, project web page, presentation and poster to showcase the project's progress and outcomes.	11	Worked on final deliverables such as DSD, Final Report, projectWeb, presentation and poster.	10	Focused on completing final deliverables, including the DSD, Final Report, project website, presentation, and poster.	10	Completed final deliverables, including the DSD report, Final Report, project website, presentation, and poster.	11	42,00
Week 18	20.01.25 - 22.01.25	Finalized the final report, DSD report, project web page, presentation and poster to showcase the project's progress and outcomes.	4	Worked on final deliverables such as DSD, Final Report, projectWeb, presentation, poster.	4	Focused on completing final deliverables, including the DSD, Final Report, project website, presentation, and poster.	4	Completed final deliverables, including the DSD report, Final Report, project website, presentation, and poster.	4	16,00
Total Effort in Man-Hours			**128,00**		**133,00**		**129,00**		**124,00**	**514,00**
Total Effort in Man-Days			**16,00**		**16,63**		**16,13**		**15,50**	**64,25**
Notes:										
1. This table shows the consolidated project effort, based on project effort tables prepared by each team member **2. Replace Team Member i with team member name and lastname, then fill out one column for each team member**										
Form: GLOWG-Project Effort Log-2024-11-10-v1.0.xlsx										

APPENDIX C3: PROJECT EFFORT LOGS FOR EACH TEAM MEMBER-

COMP 4910/4920 Project Effort Log, Project Code: GLOWG, 22.01.2025, v1.0
Team Member: Gizem Tanış

Week	Dates	Work Done in Some Detail	Total Hours Spent
Week 1	30.09.24 - 04.10.24	Contributed to team formation by defining project goals and discussing potential challenges with team members to clarify the project direction.	4
Week 2	07.10.24 -11.10.24	Participated in the project topic selection by discussing identified problems and clarifying the project scope. Assisted in consulting with advisor teachers and selecting team leads for key areas.	4
Week 3	14.10.24 - 18.10.24	Contributed to defining the project's goals and objectives, assisting in the creation of a draft proposal form to outline the initial project framework.	5
Week 4	21.10.24 - 25.10.24	Assisted in finalizing project goals and contributed to refining the draft proposal. Helped prepare and complete the PAF documentation for submission.	5
Week 5	28.10.24 - 01.11.24	Researched API integration options and potential limitations for implementing ChatGPT in a skincare context. Contributed insights from related projects to support integration planning.	6
Week 6	04.11.24 - 08.11.24	Conducted research on skincare science and AI-driven personalization to support project goals. Assisted in reviewing academic sources and identifying relevant technologies for the design phase.	6
Week 7	11.11.2024-15.11.2024	Developed a preliminary framework for a skincare recommendation system, focusing on defining the user interface and core functionalities based on the insights from similar applications and literature reviews.	6
Week 8	18.11.2024-22.11.2024	Midterm exams week	0
Week 9	25.11.24 -29.11.24	Evaluated the feasibility of proposed features and contributed to aligning project components with overall objectives.	6
Week 10	02.12.24 - 06.12.24	Researched database optimization techniques and security measures to ensure data integrity and protection. Contributed to identifying potential challenges and solutions for smooth implementation.	8
Week 11	09.12.24 - 13.12.24	Focused on researching and designing the database structure to ensure efficient data management for the Glow Genie project.	7
Week 12	16.12.24 - 20.12.24	Focused on researching database optimization techniques and began creating UML diagrams, including class and entity-relationship diagrams, to design the data architecture for the project.	7
Week 13	23.12.24 - 27.12.24	Refined the database design and UML diagrams based on further research, focusing on optimizing the data architecture for the project.	9
Week 14	30.12.24 - 03.01.25	This week, significant changes were made within the scope of the project, and	24

		the Requirement Specification Document (RSD) was reorganized to adapt to the updated goals. I conducted research to design accurate skin type test questions and answer options for the machine learning model. A survey was prepared and organized for data collection. I created UML diagrams for some of the project's functions and added detailed explanations for each diagram. Additionally, I contributed to the design of the user interface and participated in the demo development process. I reviewed and redesigned the database table structures, implementing necessary improvements to optimize data flow. I also reviewed the overall system flow to improve the project's progress and ensure smooth operation.	
Week 15	06.01.25 - 10.01.25	This week, I finalized the UML diagrams for the Requirements Specification Document (RSD) and made adjustments. We then transitioned to the Design Specification Document (DSD) and added new content. I also researched database design, focusing on table relationships, optimization techniques, security, data integrity, and backup strategies.	12
Week 17	13.01.2025 - 17.01.25	Completed final deliverables, including the DSD report, Final Report, project website, presentation, and poster.	11
Week 18	20.01.25 - 22.01.25	Completed final deliverables, including the DSD report, Final Report, project website, presentation, and poster.	4
Total Effort in Man-Hours			**124,00**

COMP 4910/4920 Project Effort Log, Project Code: GLOWG, 09.11.2024, v1.0
Team Member: Ece Topuz

Week	Dates	Work Done in Some Detail	Total Hours Spent
Week 1	30.09.2024 - 04.10.2024	A suitable team was formed for the project and organization was ensured by distributing tasks among team members.	4
Week 2	07.10.2024-11.10.2024	Defining a problem based on a detected problem and selecting the project topic accordingly.	5
Week 3	14.10.2024 - 18.10.2024	Project goals and objectives were determined and a draft project proposal was prepared.	4
Week 4	21.10.2024 - 25.10.2024	The project proposal draft was detailed and the PAF file was completed and delivered.	5
Week 5	28.10.2024 - 01.11.2024	Similar projects in the relevant field were examined and information that could be a reference for the project was collected.	6
Week 6	04.11.2024 - 08.11.2024	A literature review on AI in health and AI-driven personalized recommendations was conducted from academic sources.	7
Week 7	11.11.2024-15.11.2024	By analyzing skin care recommendation systems, similar systems and their place in the literature were examined.	6
Week 8	18.11.2024-22.11.2024	Midterm exams week	
Week 9	25.11.2024-29.11.2024	Organizing how the project's components will interact and assessing how well they work together.	7
Week 10	2.12.2024-6.12.2024	Studies were conducted on the solutions we obtained in the literature reviews, including determining its architecture, feature selection, and overall approach to ensure that it was aligned with the project's goals and requirements.	8
Week 11	09.12.24 - 13.12.24	Designed the functionality and user interaction flow of the project, emphasizing the smooth integration of components and putting forward creative interface concepts to enhance usability.	8
Week 12	16.12.24 - 20.12.24	Conducted research on appropriate machine learning algorithms and started drafting UML diagrams to represent system structure and processes.	7
Week 13	23.12.24 - 27.12.24	Refined the UML diagrams and requirements specification based on further research, finalizing the system's entry and registration mechanism.	9
Week 14	30.12.24 - 03.01.25	Significant changes were made to the project, leading to a reorganization of the RSD to align with updated goals. New functions were defined and explained in detail. Research was conducted to design skin type test questions for the machine learning model, and a survey was organized for data collection. UML diagrams were created for specific functions with detailed explanations, and a demo of the user interface was developed and included in the process.	27
Week 15	06.01.25 - 10.01.25	The RSD report was meticulously reviewed, missing functions and UML diagrams were completed and added to the report. In addition, significant progress was made within the scope of the DSD report, and detailed studies were carried out in the analysis and design stages. In this direction, the reports were made more comprehensive and understandable during the project development process.	13

Week 17	13.01.2025 - 17.01.25	Focused on completing final deliverables, including the DSD, Final Report, project website, presentation, and poster.	10
Week 18	20.01.25 - 22.01.25	Focused on completing final deliverables, including the DSD, Final Report, project website, presentation, and poster.	4
Total Effort in Man-Hours			**130,00**

COMP 4910/4920 Project Effort Log, Project Code: GLOWG, 09.11.2024, v1.0
Team Member: İrem Demir

Week	Dates	Work Done in Some Detail	Total Hours Spent
Week 1	30.09.2024 - 04.10.2024	Problem research: Define project problem, objectives, and scope. Creation of the team.	4
Week 2	07.10.2024 - 11.10.2024	Problem definition: Project selection. Clearly define the project scope.	4
Week 3	14.10.2024 - 18.10.2024	Deciding project goals: Determined the project's goals and objectives, and created a preliminary draft of the project proposal form.	5
Week 4	21.10.2024 - 25.10.2024	PAF submission: Project goals determined and prepared the PAF documentation.	5
Week 5	28.10.2024 - 1.11.2024	ChatGPT backend integration research: Investigate API documentation, integration requirements, and limitations for ChatGPT in a skincare context.	6
Week 6	4.11.2024 - 8.11.2024	Literature review: Research academic sources on skincare science, AI in health, and personalized recommendations.	6
Week 7	11.11.2024 - 15.11.2024	Analysis of similar projects: Review existing skincare recommendation systems and identify gaps to address.	6
Week 8	18.11.2024 - 22.11.2024	Midterm exams week	0
Week 9	25.11.2024 - 29.11.2024	Identified gaps in our current approach, and brainstormed innovative features to enhance our system. Explored the technical feasibility of these ideas to ensure they can be effectively implemented.	7
Week 10	02.12.2024 - 06.12.2024	Researching the requirements for integrating the GPT API into our system. Explored the technologies needed for seamless integration, including backend frameworks and libraries, and analyzed which programming language would be most suitable for ensuring compatibility with machine learning workflows and the GPT API.	8
Week 11	09.12.2024 - 13.12.2024	Researching the backend structure for the Glow Genie project. My focus was on understanding the necessary technologies, tools, and design patterns to create a robust and scalable backend system	9
Week 12	16.12.2024 - 20.12.2024	Researching UML diagrams to determine which ones would be most suitable for the project. I started designing UML diagrams, including use case, class, and activity diagrams. Additionally, as a team, we roughly sketched the project interface to visualize the overall structure.	7
Week 13	23.12.2024 - 27.12.2024	Working on the UML diagrams, refining the designs to ensure they accurately represent the system's structure and functionality. Conducted research to determine what components and features should be included in the project interface. Planning a demo version of the interface, outlining the key elements and their placement.	11

Week 14	30.12.2024 - 03.01.2025	This week, due to major changes in the Glow Genie project, I revised all our forms and UML diagrams to reflect the new direction and updated functionality. I redefined key functions, adjusted the scope to align with the project's goals, and contributed to redesigning how the system components interact. Additionally, I researched and integrated updates to the backend technologies to improve system performance and ensure seamless operation with the revised requirements. I also worked on updating user interface designs for improved usability and created detailed UML diagrams to clarify system workflows, ensuring the project remains cohesive and on track.	22
Week 15	06.01.2025 - 10.01.2025	Some missing functions have been added to the RSD report. Focused on completing the missing UML diagrams for project, ensuring that all necessary diagrams were finalized. Also prepared for the DSD report by organizing the required content and began drafting its initial sections. Additionally, worked on creating packages for the backend structure to enhance organization.	13
Week 17	13.01.2025 - 17.01.2025	Finalized the final report, DSD report, project web page, presentation and poster to showcase the project's progress and outcomes.	11
Week 18	20.01.2025 - 22.01.2025	Finalized the final report, DSD report, project web page, presentation and poster to showcase the project's progress and outcomes.	4
Total Effort in Man-Hours			**128,00**

COMP 4910/4920 Project Effort Log, Project Code: GLOWG, 10.11.2024, v1.0
Team Member: Ceren Sude Yetim

Week	Dates	Work Done in Some Detail	Total Hours Spent
Week 1	30.09.24 - 04.10.24	Formed a team for the project and defined some problems to decide on the topic of the project.	4
Week 2	07.10.24 - 11.10.24	Consulted some of the advisor teachers for the project and chose leads for main areas of the project.	4
Week 3	14.10.24 - 18.10.24	Decided on goals of the project and it's objectives and prepared a draft project proposal form.	5
Week 4	21.10.24 - 25.10.24	Worked on the draft project proposal form and readied the final version of PAF.	5
Week 5	28.10.24 - 01.11.24	Conducted research on integrating ChatGPT into the project by examining similar projects to gain insights into successful implementation strategies.	6
Week 6	04.11.24 - 08.11.24	Outlined key tasks for the design process and conducted research on potential technologies, selecting those most suitable for the project. Continued examining similar projects to gain further insights.	6
Week 7	11.11.24 - 15.11.24	Conducted research on technologies utilized in similar projects to identify and evaluate the most suitable options for implementation in our project.	5
Week 8	18.11.24 - 22.11.24	Midterm Week	0
Week 9	25.11.24 -29.11.24	Planning the interactions between the components of the project and evaluating their compatibility with one another.	7
Week 10	02.12.24 - 06.12.24	Designing the structure of the machine learning model, including determining its architecture, feature selection, and overall approach to ensure alignment with the project's objectives and requirements.	8
Week 11	09.12.24 - 13.12.24	Planned the project's functionality and user interaction flow, ensuring seamless integration of components and proposing interface ideas to enhance usability.	9
Week 12	16.12.24 - 20.12.24	Researched suitable machine learning algorithms, defined their integration strategy, and developed UML diagrams to represent the system's architecture and processes.	7
Week 13	23.12.24 - 27.12.24	Based on further research, I refined the UML diagrams and requirements specification document and finalized the machine learning algorithm for the project.	10
Week 14	30.12.24 - 03.01.25	The project underwent significant changes, so I revised the Requirement Specification Document (RSD) to align with the updated direction. I introduced new functions for the project, finalized how the machine learning models will operate, and defined the structure of the dataset. Additionally, I researched skin type characteristics to design accurate skin type test questions and answer choices, helped editing the survey we made to gather data. I also created UML diagrams for some of the functions, designed interfaces for some features, and worked on improving the overall project flow.	32

Week 15	06.01.25 - 10.01.25	Existing functions were reviewed and refined, and a new function was added. UML diagrams were created to represent the refined and newly implemented functions, providing a clearer understanding of their structure and interactions. Additionally, significant progress was made on the Data Structure Design (DSD).	11
Week 17	13.01.25 - 17.01.25	Worked on final deliverables such as DSD, Final Report, WebApp, presentation and poster.	10
Week 18	20.01.25 - 22.01.25	Worked on final deliverables such as DSD, Final Report, WebApp, presentation, poster.	4
Total Effort in Man-Hours			**133,00**