Simplify : An AI-powered Skincare Web App

By

Hein Zaw, Minji Kim, Dowon Kim

hein.zaw@stonybrook.edu
dowon.kim@stonybrook.edu
minji.kim.6@stonybrook.edu

Department of Computer Science, SUNY Korea

CSE 416: Software Development

Professor Yoonseok Yang

June 5, 2025

Targeted Audience: Skincare novices, beauty enthusiasts, busy professionals, individuals with sensitive skin, and tech-savvy consumers seeking personalized, AI-powered skincare solutions

TABLE OF CONTENTS

1. Introduction

1.1 Product Description

Skincare is very important at any time for everyone who wants to enhance one's lifestyle. Although it seems easy to just follow the basics of skin care, beauty products are enhanced to the point where each product is curated for different skin types which would not help if used on other skin types. However, people who are unaware of what to do or time-constrained by their busy schedules can not carefully construct and plan a skincare routine that is healthy for one's skin.

Our solution is to create an application that can assist people with their skincare-related problems. We will use AI image-recognition technologies to analyze each user's skin and recommend proper self-care routines. The user can also ask questions about any skincare-related topics and receive trustworthy information from the AI chatbot. With these features, users would be able to get fast and reliable guidance for each individual's needs.

Simplify is an AI-powered self-care assistant designed to streamline skincare routines for everyone. Our solution integrates **AI-driven product recommendations, an AI Skin Lab for personalized analysis, and gamification** to enhance engagement by encouraging consistency through competition and rewards. The project also explores a **win-win sponsorship model**, enabling brands to better connect with consumers in the self-care space.

1.2 Scope

The product will assist people in skincare that fits each user's skin type. The product would provide skin analysis, routines made from the information of their skin, and a streak system that could encourage the user's consistency. It will serve people who are new to skincare, people who want to take care of their skin better but do not know where to start, and people who do not have time to go through the details of skincare but want a better routine. This product would be generalizable as even if a user is knowledgeable in skincare, this would help the user to maintain their routine and assistance so that they do not have to worry about the minimal tasks of a routine. The product would run on both mobile and on the web for more accessibility for the users.

1.3 Users

Novices in skin care who have little to no knowledge about beauty but want to start changing their lifestyle would be our main target audience. However, the audience extends to beauty experts who want data-driven solutions to enhance their skin. They would need basic technical expertise such as how to take a selfie, how to use the internet, and how to navigate through a website. The ages could range from 15~60 and there is no constraint on gender, country, ethnicity, etc.

- **Novices in Skincare** – Little to no knowledge about beauty routines, seeking a simple and guided way to start.
- **Beauty Enthusiasts & Experts** – Familiar with skincare but looking for **data-driven insights** to refine their routine.
- **Varied Technical Expertise** – Includes both tech-savvy users and those less familiar with digital tools.
- **Accessibility Considerations** – Features like **adjustable text sizes, and clear navigation** for inclusivity.

1.4 Existing Alternatives

Several AI-powered dermatology services such as **lumino.skin**, **skinalyzeai.com**, and **revieve.com** have emerged, offering users skin analysis and product recommendations based on facial imaging and AI assessments. While these platforms provide a solid foundation for digital skincare consultations, they often focus heavily on cosmetic concerns and product suggestions rather than delivering in-depth insights into overall skin health. Additionally, their recommendations tend to follow generic patterns with limited adaptation to individual lifestyle, environment, or medical history. In contrast, our approach prioritizes **true personalization and broader skin care support**—not only identifying conditions but also providing tailored routines and holistic solutions grounded in each user's unique profile.

2. Requirements

2.1 Functional Requirements

1. Users must be able to receive a personalized visual skin analysis
2. Users must be able to keep track of the process by using the checklist
3. Users must be able to get AI-driven product recommendations
4. Users must be able to ask skin care related questions for instant guidance
5. Users should be able to get rewards with consistent usage

2.2 Use Cases

Use Case 1	Skin Lab Photo Analysis
Primary Actor	User
Priority	Essential
Scenario	1. The user logs into the application. 2. The user clicks on the skin lab tab on the sidebar. 3. The skin lab page should show default information of how the analysis will come out with a button to start the analysis. 4. The user clicks on the start button and asks the user for device camera access. 5. Skin lab page shows a web camera view for the user to take a photo along with buttons to take the photo and upload a photo. 6. The user clicks on the take photo button and the web camera view will turn to a still picture the user just took. 7. The user clicks on the submit button and the skin lab page will display the analysis of the user's photo.
Extensions	1a) Password is not correct 1a 1) Show error and back to sign up page 5a) User denies camera access 5a 1) The skin lab page blacks out the photo frame and waits for the user's permission

Use Case 2	Inquire about the AI assistant about skin care related problems
Primary Actor	User
Priority	Essential
Scenario	1. The user logs into the application. 2. The user clicks on the Chat tab in the sidebar. 3. The user types their skin care question in the input field. 4. The user sends the message. 5. The system replies with the appropriate answer.
Extensions	1a) Password is not correct 1a 1) Show error and back to sign up page

Use Case 3	Check the rank on the leaderboard
Primary Actor	User
Priority	Essential
Scenario	1. The user logs into the application. 2. The user navigates to the leaderboard section from the sidebar. 3. The system displays the leaderboard showing ranks of all users based on their skincare routine consistency. 4. The user locates their rank, which is highlighted or listed with their points, streaks, and rewards.
Extensions	1a) Password is not correct 1a 1) Show error and back to sign up page

Use Case 4	Finish the Routine
Primary Actor	User
Priority	High
Scenario	1. The user logs into the application and navigates to the routine page. 2. The user sees a list of tasks in their skincare routine for the day. 3. The user completes each task in the routine (e.g., applying cleanser, moisturizer, etc.). 4. The user checks the checkbox next to each completed task in the routine. 5. Once all tasks are completed, the user checks the final finish button to mark the routine as finished. 6. The system records the completion and adds to the user's streak count and score. 7. The homepage keeps the routine checkboxes checked so there is no repetition of finishing the routine.
Extensions	1a) Password is not correct 1a 1) Show error and back to sign up page 5a) User has already completed the routine but tries to mark it again 5a 1) The system will display an error message and remind them that the routine has already been finished for the day.

Use Case 5	Delete the user's data
Primary Actor	Admin
Priority	Essential
Scenario	1. The admin logs into the system with proper credentials. 2. The admin navigates to the user management section. 3. The admin selects the user whose data needs to be deleted. 4. The admin confirms the deletion request. 5. Upon confirmation, the system securely deletes the user's personal data, including skincare preferences, recommendations,

	and any stored images or analysis.
Extensions	6a. If a server error occurs during the data deletion, the system will display an error message and prompt the admin to try again.

2.3 Non-functional Requirements

1. Security:

- User data (e.g., skin analysis results) must be encrypted using HTTPS.
- Basic authentication (email/password) for user login.
- 2FA with Google OAuth

2. Usability:

- The app should be intuitive, with simple navigation requiring minimal guidance (less than 10 minutes for a new user to get started).
- Clear error messages for invalid input or incomplete steps.

3. Compatibility:

- The app should work on the latest versions of Chrome and Firefox.
- It should be responsive for screens from phones, tablets and desktops.

4. Maintainability:

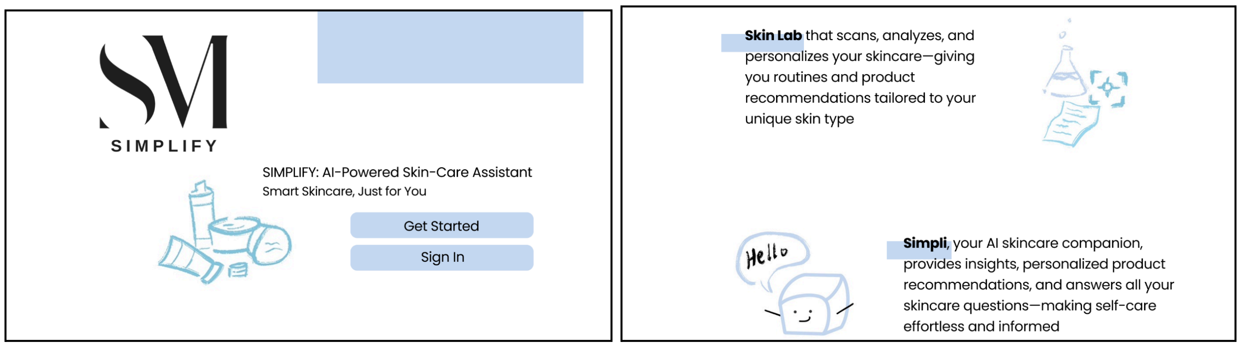
- The code should be modular and easy to update (less than 50 lines per function).
- Use basic automated tests to check app functionality (e.g., test login and skin analysis features).

5. Performance:

- AI routine recommendations must be generated in no more than 60 seconds
- Chatbot responses should be generated in no more than 10 seconds

3. Design

3.1 User Interfaces



SIMPLIFY

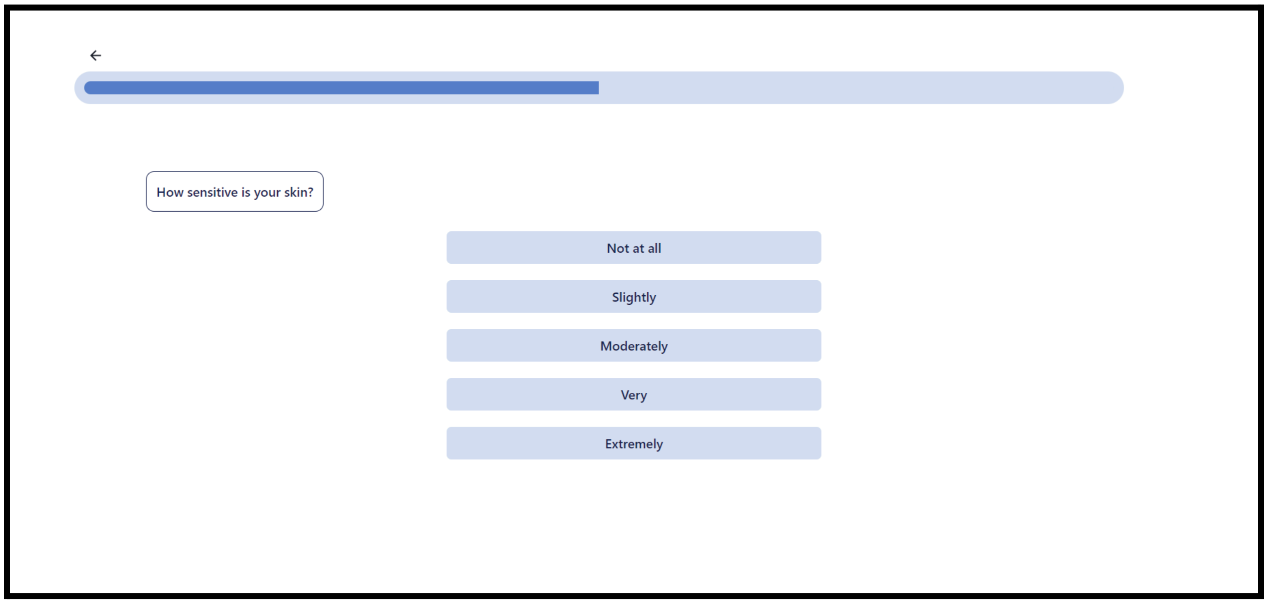
SIMPLIFY: AI-Powered Skin-Care Assistant
Smart Skincare, Just for You

Skin Lab that scans, analyzes, and personalizes your skincare—giving you routines and product recommendations tailored to your unique skin type

Simpli, your AI skincare companion, provides insights, personalized product recommendations, and answers all your skincare questions—making self-care effortless and informed

Fig 1: Welcome page

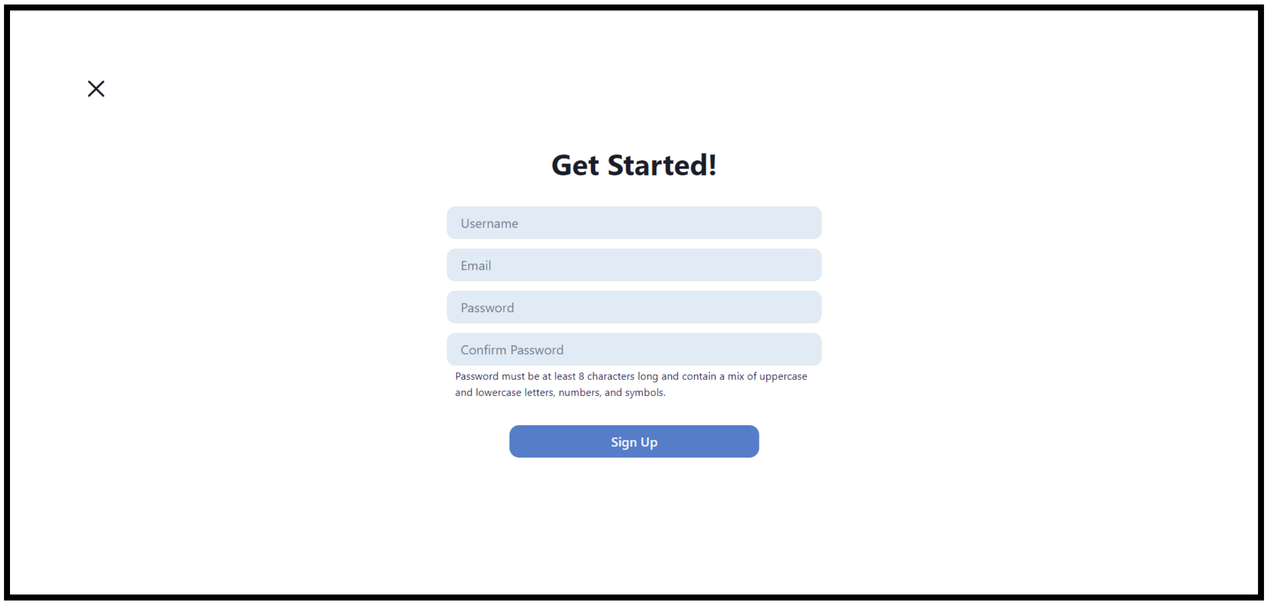
- Shows the description of the features and what to expect on the website.
- Users can get started or sign in.



How sensitive is your skin?

Fig 2: Personalized Survey

- Users will answer a survey as they sign up so the application can provide personalized information from the start.



Password must be at least 8 characters long and contain a mix of uppercase and lowercase letters, numbers, and symbols.

Fig 3: Sign Up Page

- Users will input their unique username along with their email and password.
- After entering all three fields, they will be able to create their account with the personalized information from before.

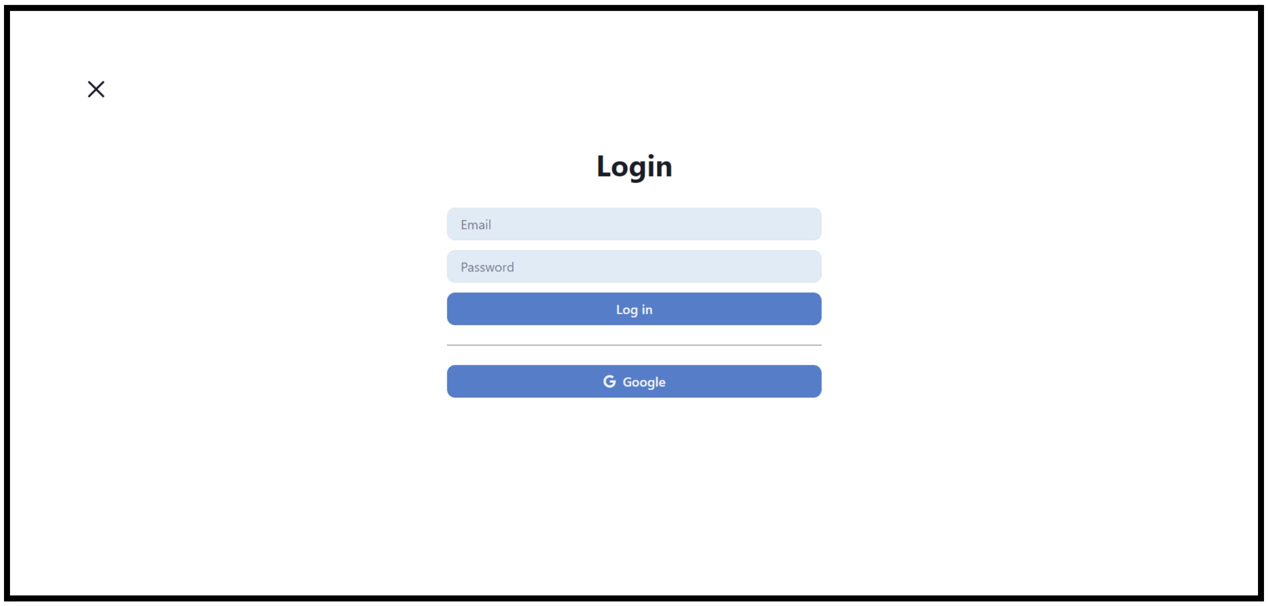
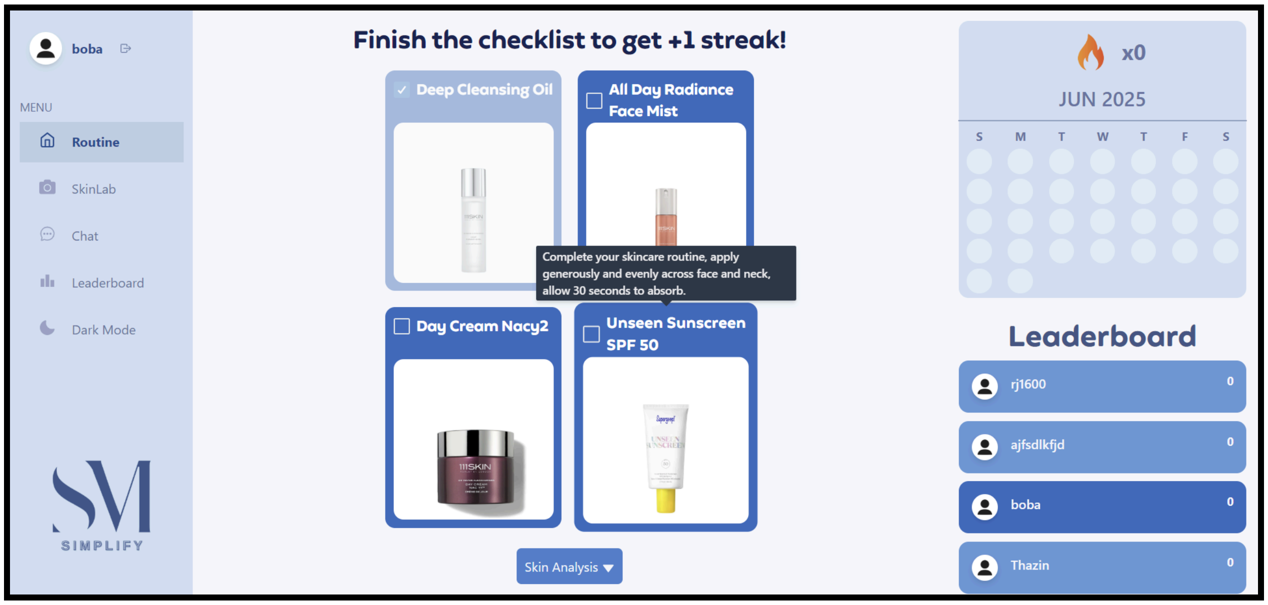


Fig 4: Login Page

- Users can login with their email and password or their Google email.
- If they do not have an account, users can go to the sign up page from the sign up button.

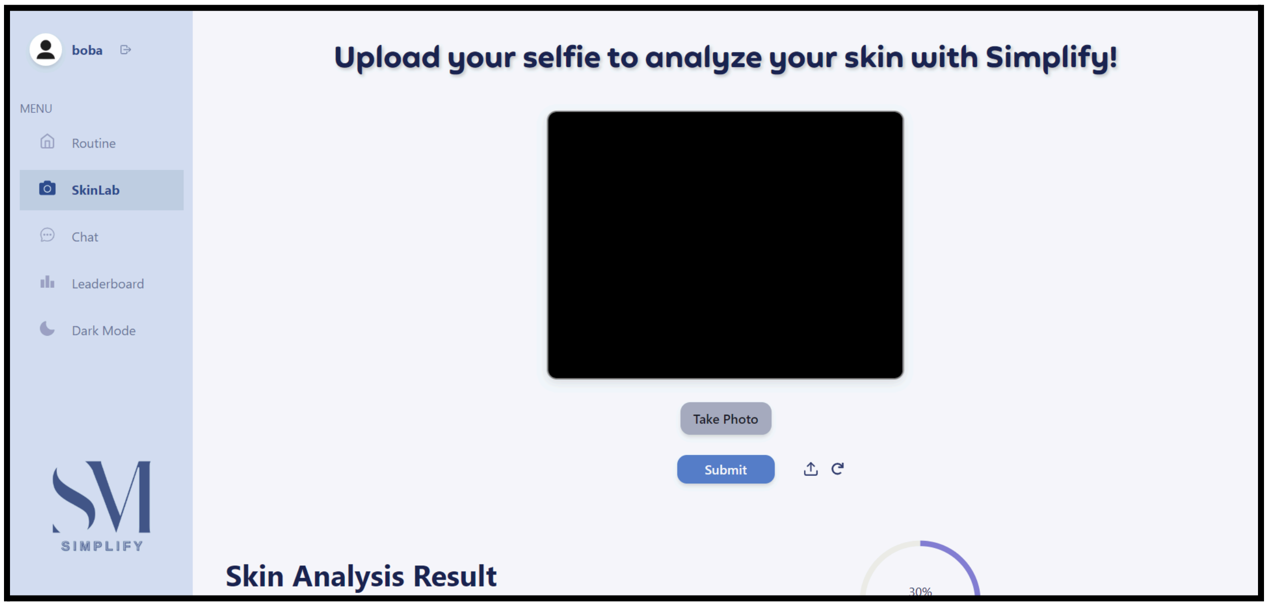


Finish the checklist to get +1 streak!

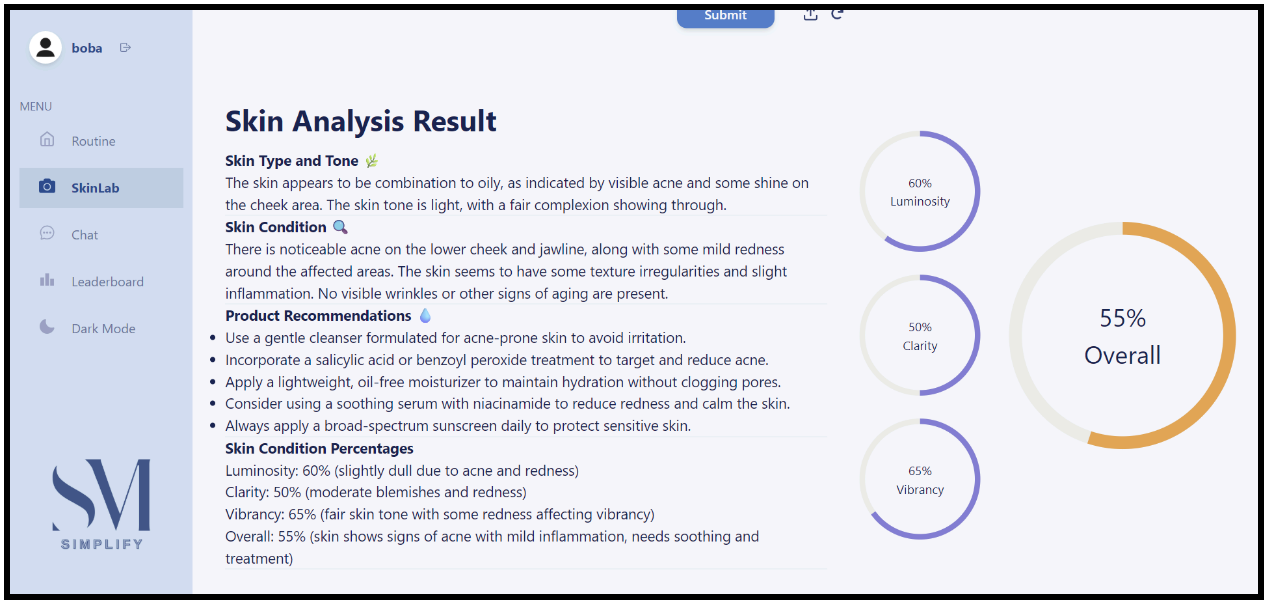
Complete your skincare routine, apply generously and evenly across face and neck, allow 30 seconds to absorb.

Fig 5: Home Page

- Shows User Recommended Routine, Streak Calendar, Brief Leaderboard, and user skin analysis



Upload your selfie to analyze your skin with Simplify!



Skin Analysis Result

Skin Type and Tone 🌿
The skin appears to be combination to oily, as indicated by visible acne and some shine on the cheek area. The skin tone is light, with a fair complexion showing through.

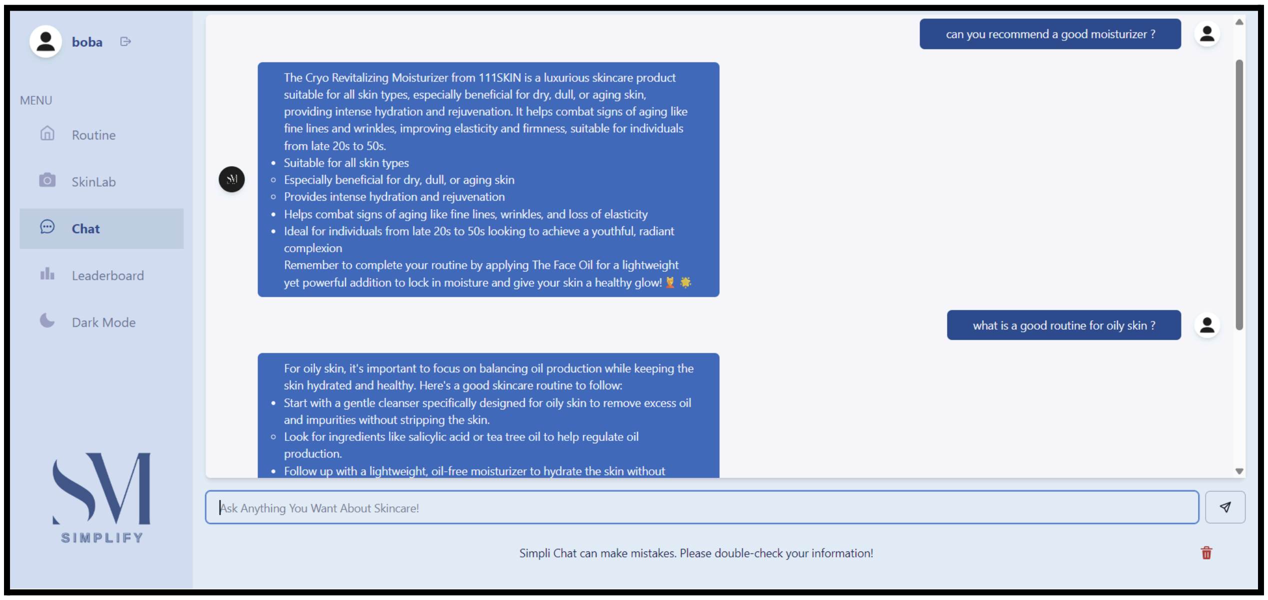
Skin Condition 🔍
There is noticeable acne on the lower cheek and jawline, along with some mild redness around the affected areas. The skin seems to have some texture irregularities and slight inflammation. No visible wrinkles or other signs of aging are present.

Product Recommendations 💧
- Use a gentle cleanser formulated for acne-prone skin to avoid irritation.
- Incorporate a salicylic acid or benzoyl peroxide treatment to target and reduce acne.
- Apply a lightweight, oil-free moisturizer to maintain hydration without clogging pores.
- Consider using a soothing serum with niacinamide to reduce redness and calm the skin.
- Always apply a broad-spectrum sunscreen daily to protect sensitive skin.

Skin Condition Percentages
Luminosity: 60% (slightly dull due to acne and redness)
Clarity: 50% (moderate blemishes and redness)
Vibrancy: 65% (fair skin tone with some redness affecting vibrancy)
Overall: 55% (skin shows signs of acne with mild inflammation, needs soothing and treatment)

Fig 6: SkinLab Page

- Users can get a personalized skin analysis by taking a photo or uploading a photo.



can you recommend a good moisturizer ?

The Cryo Revitalizing Moisturizer from 111SKIN is a luxurious skincare product suitable for all skin types, especially beneficial for dry, dull, or aging skin, providing intense hydration and rejuvenation. It helps combat signs of aging like fine lines and wrinkles, improving elasticity and firmness, suitable for individuals from late 20s to 50s.
- Suitable for all skin types
  - Especially beneficial for dry, dull, or aging skin
  - Provides intense hydration and rejuvenation
- Helps combat signs of aging like fine lines, wrinkles, and loss of elasticity
- Ideal for individuals from late 20s to 50s looking to achieve a youthful, radiant complexion

Remember to complete your routine by applying The Face Oil for a lightweight yet powerful addition to lock in moisture and give your skin a healthy glow! 💆 🌟

what is a good routine for oily skin ?

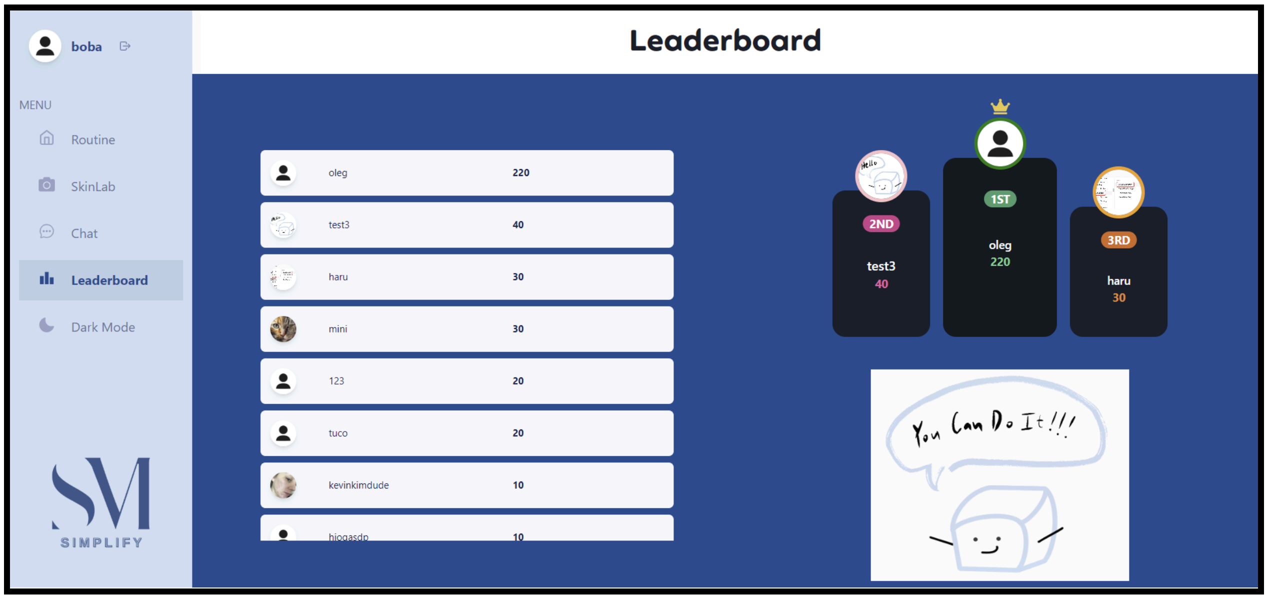
For oily skin, it's important to focus on balancing oil production while keeping the skin hydrated and healthy. Here's a good skincare routine to follow:
- Start with a gentle cleanser specifically designed for oily skin to remove excess oil and impurities without stripping the skin.
  - Look for ingredients like salicylic acid or tea tree oil to help regulate oil production.
- Follow up with a lightweight, oil-free moisturizer to hydrate the skin without

Ask Anything You Want About Skincare!

Simpli Chat can make mistakes. Please double-check your information!

Fig 6: Chat With Simpli

- Users can chat with Simpli to get answers to their skincare related questions.



oleg	220
test3	40
haru	30
mini	30
123	20
tuco	20
kevinkimdude	10
hjogasdp	10

Fig 7: Leaderboard Page

- Users can see the leaderboard categorized by cumulative/monthly and regional/global. Which has been updated to the leaderboard of just cumulative scores.

3.2 Data Design

UML Data Diagram contains classes User, Product, Feedback, and Chat.
User has referenced attributes to a Feedback, an array of Product, and an array of Chat.

LucidChart Data Diagram

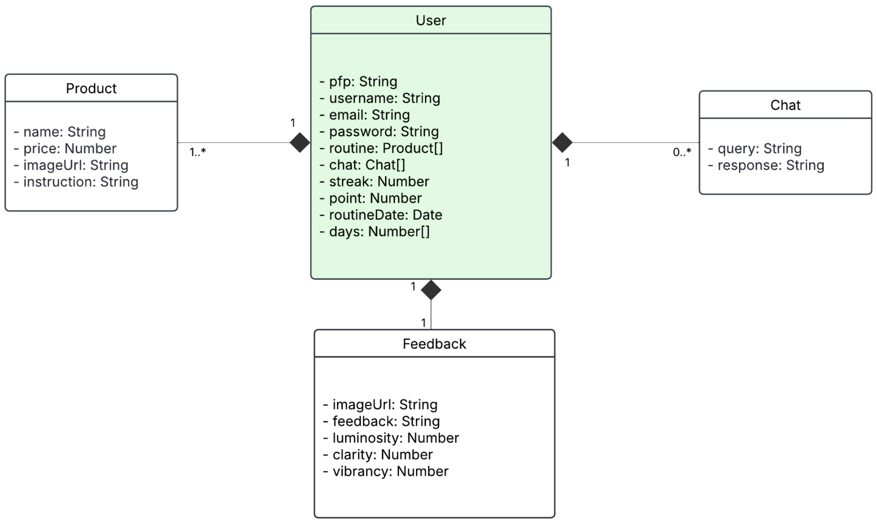
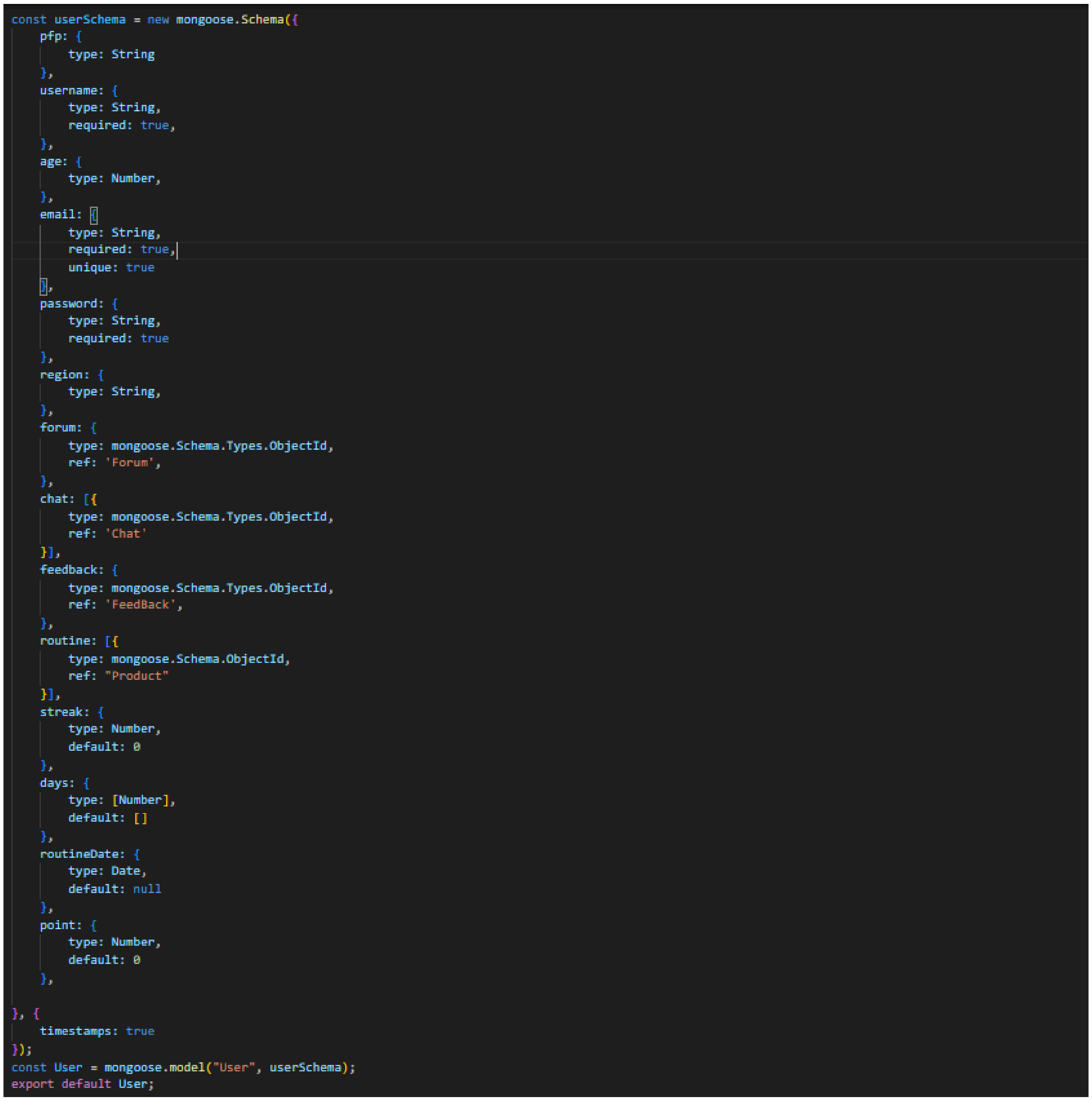


Fig 1: UML Data Diagram

MongoDB Data Diagram

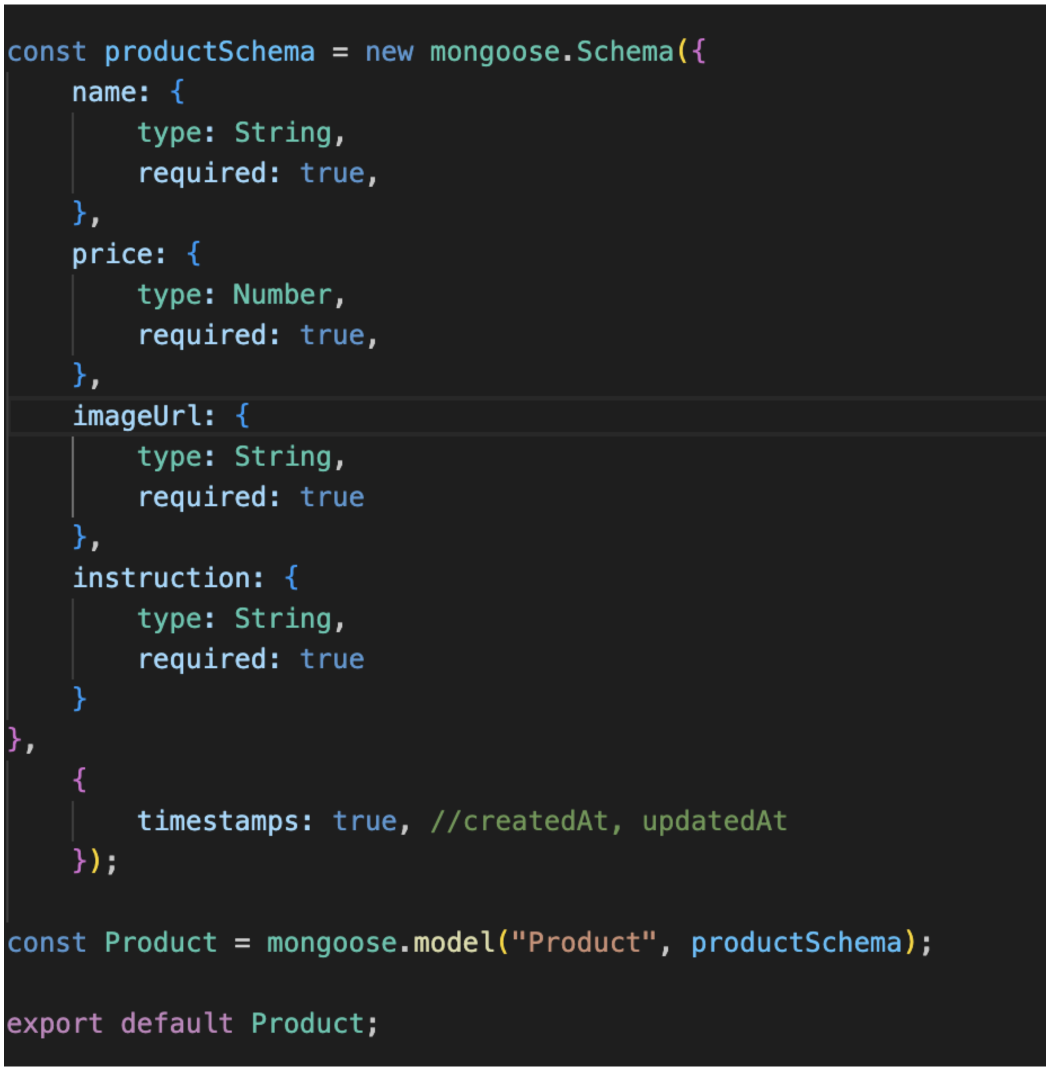
We used ODM(Object Document Mapping) to map our classes to the database using MongoDB schemas



```
const userSchema = new mongoose.Schema({
    pfp: {
        type: String
    },
    username: {
        type: String,
        required: true,
    },
    age: {
        type: Number,
    },
    email: {
        type: String,
        required: true,
        unique: true
    },
    password: {
        type: String,
        required: true
    },
    region: {
        type: String,
    },
    forum: {
        type: mongoose.Schema.Types.ObjectId,
        ref: 'Forum',
    },
    chat: [{
        type: mongoose.Schema.Types.ObjectId,
        ref: 'Chat'
    }],
    feedback: {
        type: mongoose.Schema.Types.ObjectId,
        ref: 'FeedBack',
    },
    routine: [{
        type: mongoose.Schema.ObjectId,
        ref: "Product"
    }],
    streak: {
        type: Number,
        default: 0
    },
    days: {
        type: [Number],
        default: []
    },
    routineDate: {
        type: Date,
        default: null
    },
    point: {
        type: Number,
        default: 0
    },

}, {
    timestamps: true
});
const User = mongoose.model("User", userSchema);
export default User;
```

Fig 2: user Schema

The user schema contains information about the user such as pfp(profile picture), username, age, region, streak. As MongoDB is not designed to store large binary files we decided to use URLs for images to keep our database lightweight and easier to manage. Region and streak will be used for our leaderboard(gamification). Email and password are related to user authentication. User schema also references forum, chat, feedback, and routine classes.

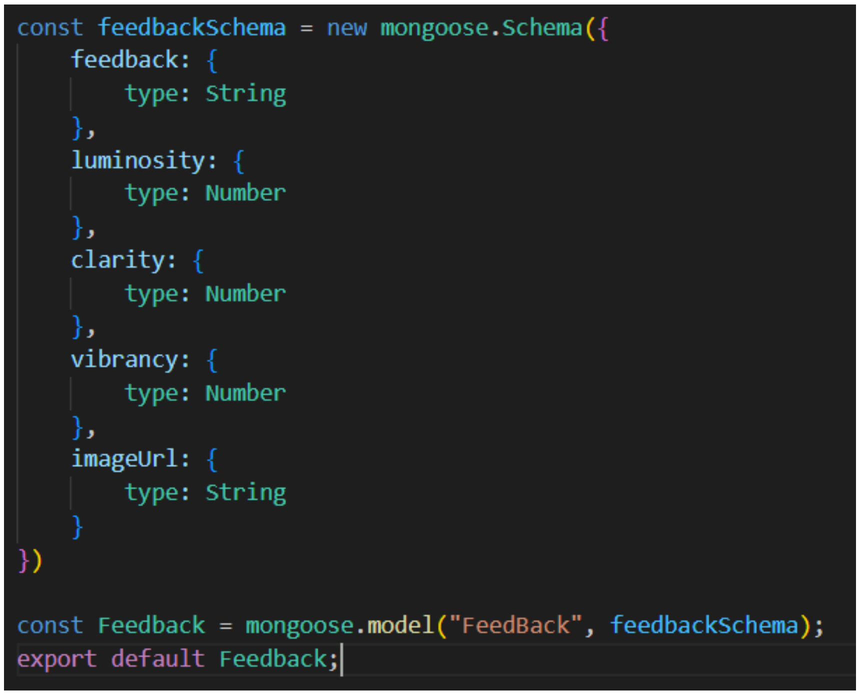


```
const productSchema = new mongoose.Schema({
    name: {
        type: String,
        required: true,
    },
    price: {
        type: Number,
        required: true,
    },
    imageUrl: {
        type: String,
        required: true
    },
    instruction: {
        type: String,
        required: true
    }
},
    {
        timestamps: true, //createdAt, updatedAt
    });

const Product = mongoose.model("Product", productSchema);

export default Product;
```

Fig 3: Product Schema

Product schema contains the product name, price, and imageURL for the information of products. Also each product has an instruction on how to use it for a user's routine.

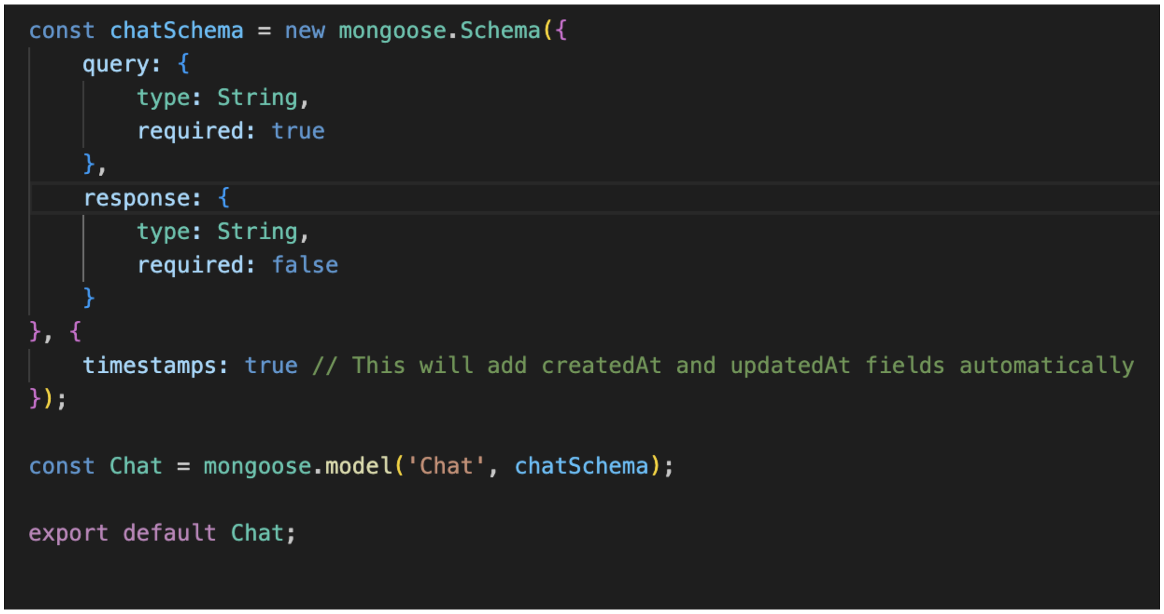
The timestamps option is enabled to automatically store createdAt and updatedAtvalues for each leaderboard document.



```
const feedbackSchema = new mongoose.Schema({
    feedback: {
        type: String
    },
    luminosity: {
        type: Number
    },
    clarity: {
        type: Number
    },
    vibrancy: {
        type: Number
    },
    imageUrl: {
        type: String
    }
})

const Feedback = mongoose.model("FeedBack", feedbackSchema);
export default Feedback;
```

Fig 6: Feedback Schema

Feedback schema is for storing the information of the Skin Labs output. It stores the text feedback luminosity, clarity, vibrancy, and skinUrl. Luminosity, clarity, and vibrancy are the metrics for the skin analysis and skinUrl is for the user's input image for skin analysis.



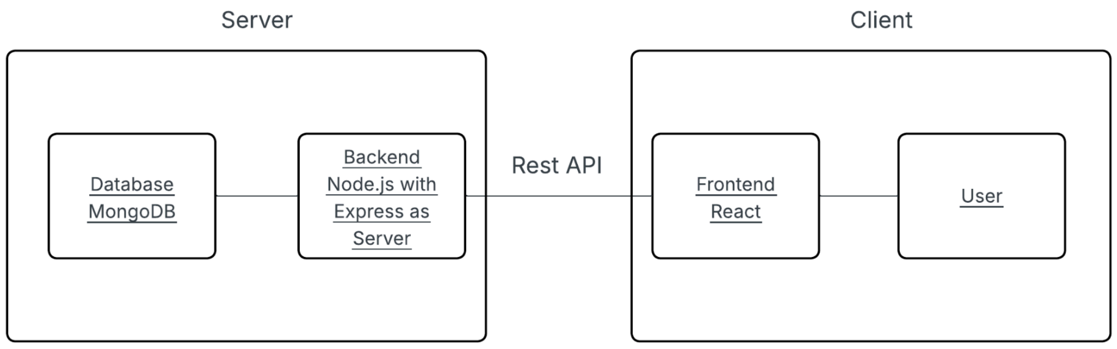
```
const chatSchema = new mongoose.Schema({
    query: {
        type: String,
        required: true
    },
    response: {
        type: String,
        required: false
    }
}, {
    timestamps: true // This will add createdAt and updatedAt fields automatically
});

const Chat = mongoose.model('Chat', chatSchema);

export default Chat;
```

Fig 7: Chat Schema

Chat schema is for storing each query and response for each user's query. By saving chats in an array, we will save the user chat history.

3.3 System Architecture

We are using a client-server architecture which is a distributed application structure that partitions tasks or workloads. With the MERN stack that uses MongoDB, Express.js, React, and Node.js. On the client side of the architecture, there will be the user and the frontend which will be React. The server side of the architecture will be the Server with Express.js and Node.js and the Database with MongoDB. For the connection of the server side and the client side, we will use REST APIs.



Since we are using the MERN stack we are primarily using Javascript-based code on both frontend and backend. We also additionally used CSS to ensure consistent styling across the application and to enhance the modularity of components, making them easier to manage and reuse.

Other than the above we are using some more third-party libraries and frameworks.

Frontend

For the frontend we are using Vite which is a helpful and fast development tool optimized for JavaScript frameworks like React. For consistency of design for the UI, we are using Chakra UI.

Major versions

- React: V19
- Vite: V6
- ChakraUI: V2

Backend and Database

Backend server is just Express.js and Node.js. Our database will be mainly MongoDB for storing user credentials and other persistent application data. In addition to MongoDB we will use Supabase which is an open-source used to store vectorized or CSV-based web scraped data. Also Cloudinary for the storing of images.

Major versions

- Express.js: V4
- MongoDB: V8

Authentication and Others

For authentication we are also planning to use Google OAuth which allows secure and token-based sign in through Google accounts. Regarding deployment will be looked into in section 3.5 of this document.

Major versions

- Google OAuth: V2

Reason for this technology stack

We chose this technology stack as the MERN stack is widely recognized as an industry-standard stack for modern web development. It also offers a unified Javascript environment across both frontend and backend simplifying the development process. Additionally, the team has prior experience with this stack through previous courses making it easier for all our team members to easily follow on productively.

3.4 Code Conventions

For code conventions, since we are primarily using Javascript on both frontend and backend, we decided to follow the Google JavaScript Style Guide. Also as we have css to support our styling, we decided to follow the Google HTML/CSS Style Guide.

For API design conventions, we used Microsoft's API design practices.

3.5 API Design

We've put all our API details in the spreadsheet linked below. We designed RESTful APIs by following the API design practices from Microsoft.

Google Sheet link: Simplify - API Design

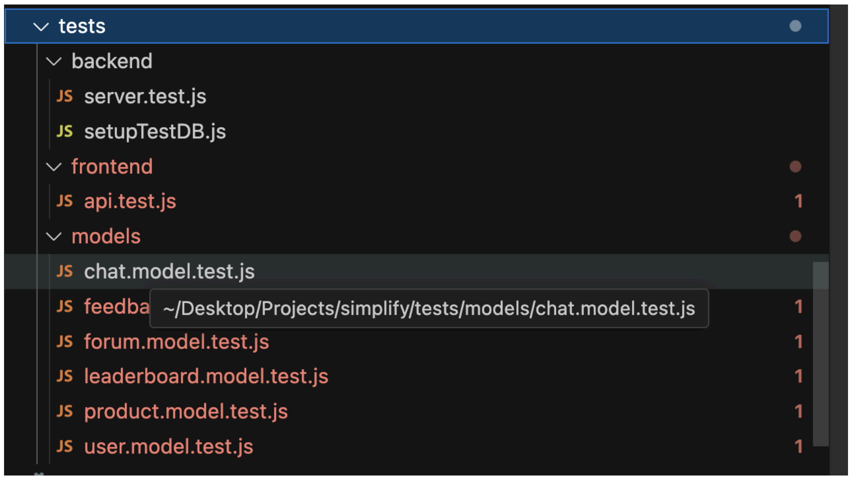
4. Testing

4.1 Unit + Integration Testing

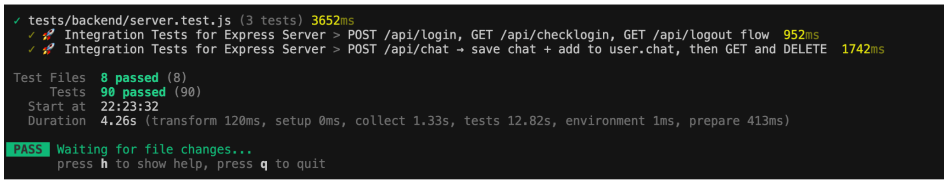
Run **npm run test** in project root folder to run the unit + integration tests.

⚠️ WARNING: Running `npm run test` will delete all users in the database except `admin@admin.com`. Make sure you're not connected to the production database when running tests.

To ensure the reliability of our application, we implemented comprehensive unit and integration testing covering the backend server, frontend API endpoints, and every schema model in our database. Unit tests were written for each Mongoose model, including user, chat, feedback, forum, product, and leaderboard, to validate the integrity of data structures and schema methods.



```
∨ tests                                                  ●
  ∨ backend
    JS server.test.js
    JS setupTestDB.js
  ∨ frontend                                             ●
    JS api.test.js                                       1
  ∨ models                                               ●
    JS chat.model.test.js
    JS feedba  ~/Desktop/Projects/simplify/tests/models/chat.model.test.js   1
    JS forum.model.test.js                               1
    JS leaderboard.model.test.js                         1
    JS product.model.test.js                             1
    JS user.model.test.js                                1
```



```
✓ tests/backend/server.test.js (3 tests) 3652ms
  ✓ 🚀 Integration Tests for Express Server > POST /api/login, GET /api/checklogin, GET /api/logout flow  952ms
  ✓ 🚀 Integration Tests for Express Server > POST /api/chat → save chat + add to user.chat, then GET and DELETE  1742ms

Test Files  8 passed (8)
     Tests  90 passed (90)
  Start at  22:23:32
  Duration  4.26s (transform 120ms, setup 0ms, collect 1.33s, tests 12.82s, environment 1ms, prepare 413ms)

 PASS  Waiting for file changes...
       press h to show help, press q to quit
```

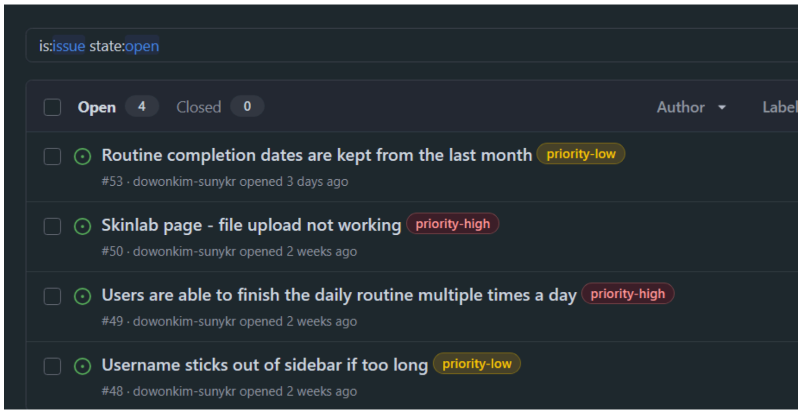
On the integration side, we tested all major backend server routes and frontend API flows such as user login/logout, authentication checks, and chat interactions. These integration tests verified that the entire request-response cycle functioned correctly across different layers of the stack. In total, 90 tests across 8 files passed successfully, demonstrating strong coverage and system stability. We also utilized a watch mode to re-run tests automatically during development, allowing for efficient debugging and continuous quality assurance.

4.2 Beta Testing

For our beta release presentation on May 21st, we had the opportunity to showcase the core functionality of our website to our classmates and receive initial feedback. After the presentation, we held a short beta testing session where our classmates interacted with the site directly, testing its main features and overall user interface. This informal testing phase helped us identify various bugs and gather suggestions for improving user experience. For example, we were informed of a critical bug that allowed one of the testers to cheat the points system by completing the routine multiple times a day, which made us prioritize fixing that bug until the final release. The beta testing session also helped confirm that key features like user authentication, AI routine generation, and chatbot responses were working as expected. Overall, it gave us valuable feedback that we used to improve our software until the final release.

5. Bug Reporting

For our bug reports, we mainly used our repository's GitHub issue page to track and manage issues collaboratively. Each bug was described in detail, including what it looks like, how to reproduce it, and the priority.



is:issue state:open

Open 4 Closed 0 Author Label

Routine completion dates are kept from the last month priority-low
#53 · dowonkim-sunykr opened 3 days ago

Skinlab page - file upload not working priority-high
#50 · dowonkim-sunykr opened 2 weeks ago

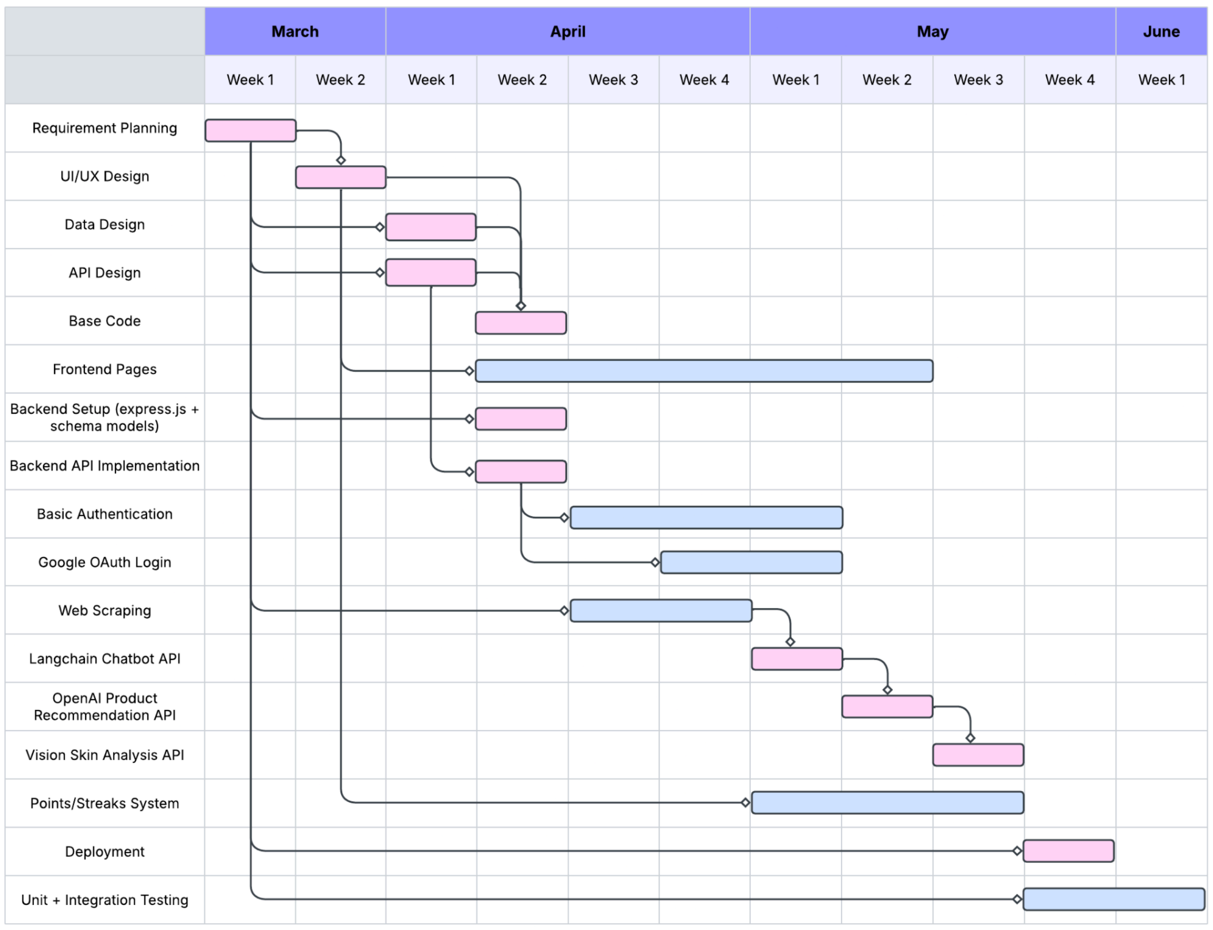
Users are able to finish the daily routine multiple times a day priority-high
#49 · dowonkim-sunykr opened 2 weeks ago

Username sticks out of sidebar if too long priority-low
#48 · dowonkim-sunykr opened 2 weeks ago

For more of our bugs, we put it on a collaborative todo list for everyone to see the progress of it.
Link: Simplify_Bug_Report

6. Schedule

Simplify Gantt Chart



Our Gantt chart shows the full timeline of our project from March to early June. We started off with requirement planning, UI/UX design, and database/API design during the first few weeks, and then moved on to backend setup and frontend page development. Tasks like setting up Express.js, MongoDB schema models, and frontend components were divided early so we could work in parallel. As the project progressed, we started integrating more complex features like the Langchain chatbot, OpenAI product recommendation, and the Vision Skin Analysis API. We also made sure to leave time at the end for deployment and unit/integration testing. Overall, the chart helped us stay organized and on track throughout the semester.

Looking back, one thing we could’ve done better is giving ourselves a bit more buffer time, especially toward the end for testing and fixing unexpected bugs. Some parts took longer than we thought, and overlapping tasks too much sometimes created bottlenecks. Next time, having clearer checkpoints or weekly goals might help us catch delays earlier and manage last-minute crunch time better.

7. Deployment

For deploying our web application we will use,

Render (for Backend/API/Frontend hosting) : https://simplify-e3px.onrender.com/

Render has a **free tier** that allows you to host web services with **auto-sleep after 15 minutes of inactivity**. It is very accessible and with its free tier and auto-sleep feature after 15 minutes of inactivity, it provides a cost-effective solution for long-term use. For our final deployment we wanted to use the cost-effective free solution. We looked into Heroku and AWS but the work was tedious so we just decided to buy $7/month payment for **final deployment.** We deploy both frontend and backend at the same port for accessibility.

8. Self Reflection and Peer Assessment

8.1 Hein Zaw (115344093)

Throughout this project, I took the lead on many of the foundational and backend-related tasks. I was responsible for the initial project setup, including configuring Supabase for auth and storage, setting up Vite for the frontend environment, ViteTest for testing, and establishing the MERN stack for full-stack development. I also integrated LangChain with the OpenAI APIs to support the AI-based routine generation, Skin Analysis and Chatbot features. I also scraped the data because I wanted my chatbot to be knowledge based, ie. Retrieval Augmented Generation (RAG). On top of that, I created and managed the MongoDB schema models for users, chats, forums, feedback, and other components. I also set up and maintained the unit and integration testing infrastructure, ensuring our backend server, API routes, and schema models were all functioning as expected. I deployed the app using Render and handled environment configurations to keep the deployment stable. While most of my focus was on the backend and AI features, I also contributed to the frontend, especially on the welcome page, chat page, and skin analysis page where I helped implement layout and connect APIs.

Team collaboration was a big part of our success, and I really appreciate how much Dowon and Minji contributed to their respective areas. Dowon played a key role in handling Google authentication, building user-related APIs, and integrating those APIs across the entire stack. He was also the go-to person for fixing bugs that affected both frontend and backend, and he worked on integrating the chat API with LangChain. Minji focused on the frontend design and user experience. She implemented most of the UI for key pages and made sure our site looked and felt polished. She also contributed to the skin analysis page and worked with me on integrating the Vision API. Minji also made color palettes, style components, light screen dark screen, state management and drawings. She was involved in almost every page developed. Both Dowon and Minji were great team players, and I'm proud of how we supported each other through the whole semester project, discussion sessions, and our final deployment. This project really helped me improve both technically and as a collaborator. If I could give up to 100 marks to both my teammates, I would give 101! I believe we all enjoyed working on this, mainly because we treated this as not the class project but our actual portfolio project for future career.

8.2 Minji Kim (115242397)

For my self reflection of this project, I think I did a great job on being there for the team and contributing to my assigned parts. My assigned parts were primarily focused on the frontend of the stack. It was because I was most interested in the frontend components and also the design of it. Therefore I did not have much time to look at the backend but I was able to work on small portions of it towards the end as it was a major part of our development. Through the whole project I was able to learn more about the management of a team which was not a prominent factor for my prior classes. It was interesting as it would need a lot of communication, documentation and organization which took up more time and thought than I expected. For the documentation we had to change some parts from our requirements which I struggled with in making sure that the changed parts integrated with the current code that we had. This showed me how important it was to have a solid understanding of what we want our app to do from the start so that there is less loss later in the process of fixing the changes. In addition I realized that I drive for perfectionism and I learned how to smoothly combine it in team settings where there are a lot of opinions. The project development with my teammates have helped me be more prepared for the collaborative work experiences I will be encountering in the future.

For my peers I am glad I have met my team as we ended up having great synergy together. At the start it was a bit rocky as we did not know each other that well. As we had meetings and discussions we were able to get comfortable and fully focus on our project. Both my teammates were great in their areas of responsibility. Henry focused on the backend regarding the databases, web scraping, and AI APIs. Dowon focused on google authentication, user related APIs, and worked on every part of the overall stack where he went back and forth for the integration and many bug fixing as well. Both Henry and Dowon were great teammates who worked hard for our collective goal of making a great project outcome. I am glad I have worked with both of them as we were all there for each other for numerous hardships met during the development process and had each other's backs for all the hard times. Thanks to such great teamwork, I believe that we were able to reach such a successful outcome which was similar to our original idea. I hope to meet such great peers later in my future projects and also make sure that I am great peers to my future teams.

8.3 Dowon Kim (114001605)

I believe I've contributed my fair share to this project and didn't let my teammates down. Since I didn't focus solely on one stack of our application, I was able to get a good understanding of both the frontend and backend, and provide quick solutions to any bugs or fixes across the entire application. This allowed me to more effectively collaborate with my team and increase our efficiency. Overall, the main lesson I learned from this project is the importance of communication. While we weren't in a professional setting for this course, I was able to practice how I present my ideas and contributions to my teammates during our numerous scrum meetings and sprints. I feel confident that these experiences have prepared me well to contribute effectively in a real professional work environment.

I would also like to say that I've had the great pleasure of meeting and working with Henry and Minji for the first time. Both of them made significant contributions by focusing on the areas where they excelled. Henry took the lead on database management and AI API integration, while Minji handled much of the frontend work and designs, which were often tedious for the rest of us and sometimes overlooked. They consistently gave their all, held meetings regularly, and genuinely cared about the success of our project. Even with all of our documents and presentations, they both worked very hard to not let our team down. I was initially slightly worried about doing a semester-long final project with classmates I had never spoken to before, but now I feel only appreciation and gratitude towards them for being great peers and friends.