Data Design & Web Views

Hein Zaw (115344093) hein.zaw@stonybrook.edu
Minji Kim (115242397) minji.kim.6@stonybrook.edu
Dowon Kim (114001605) dowon.kim@stonybrook.edu

How To Run

1. Git clone git@github.com:HeinHtutZaw19/simplify.git
2. Create a .env file in project root folder, and copy these keys

```
PORT = 4000
MONGO_URI =
"mongodb+srv://heinzaw:69gQJgesRPloWbhJ@simplify.i7eqo.mongodb.net/?retryWrites=true&w=majority&appName=Simplify"
OPENAI_API_KEY =
"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"
SUPABASE_URL_CHATBOT = "https://gsklwlfjuenpokozvlot.supabase.co"
SUPABASE_API_KEY =
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJzdXBhYmFzZSIsInJlZiI6Imdza2x3bGZqdWVucG9rb3p2bG90Iiwicm9sZSI6ImFub24iLCJpYXQiOjE3NDI5NzAxODUsImV4cCI6MjA1ODU0NjE4NX0.Jx3OIIyCpF9BjofzjokkVml8_SN6ppkRIpykQAdFmTk"å
```

2. npm install
3. cd frontend
4. npm install
5. cd ..
6. npm run dev
7. cd frontend
8. npm run dev
9. Access http://localhost:5173

Data Diagram

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

LucidChart Data Diagram

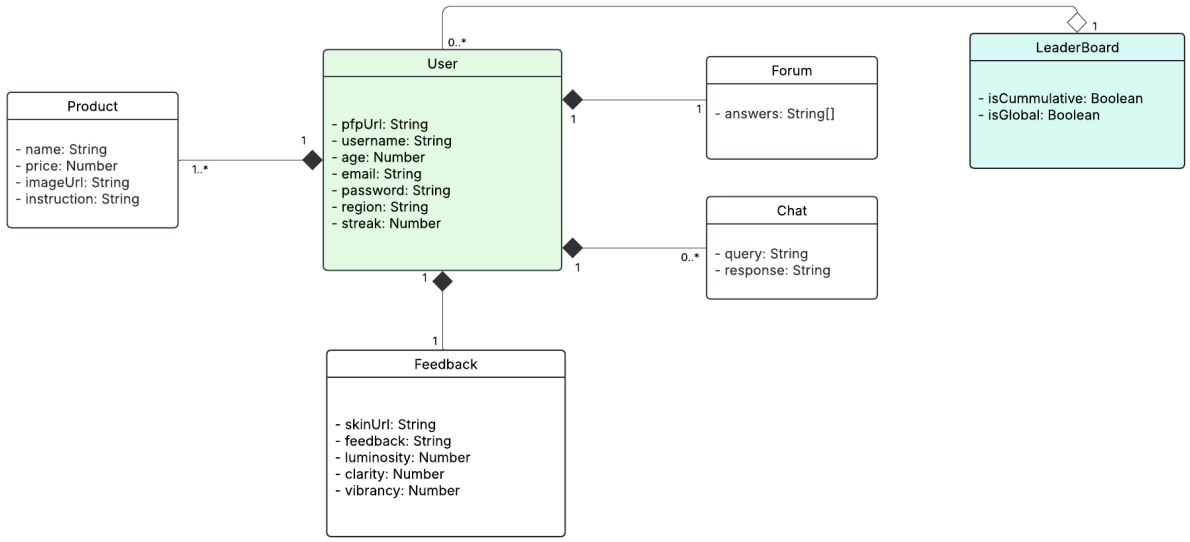
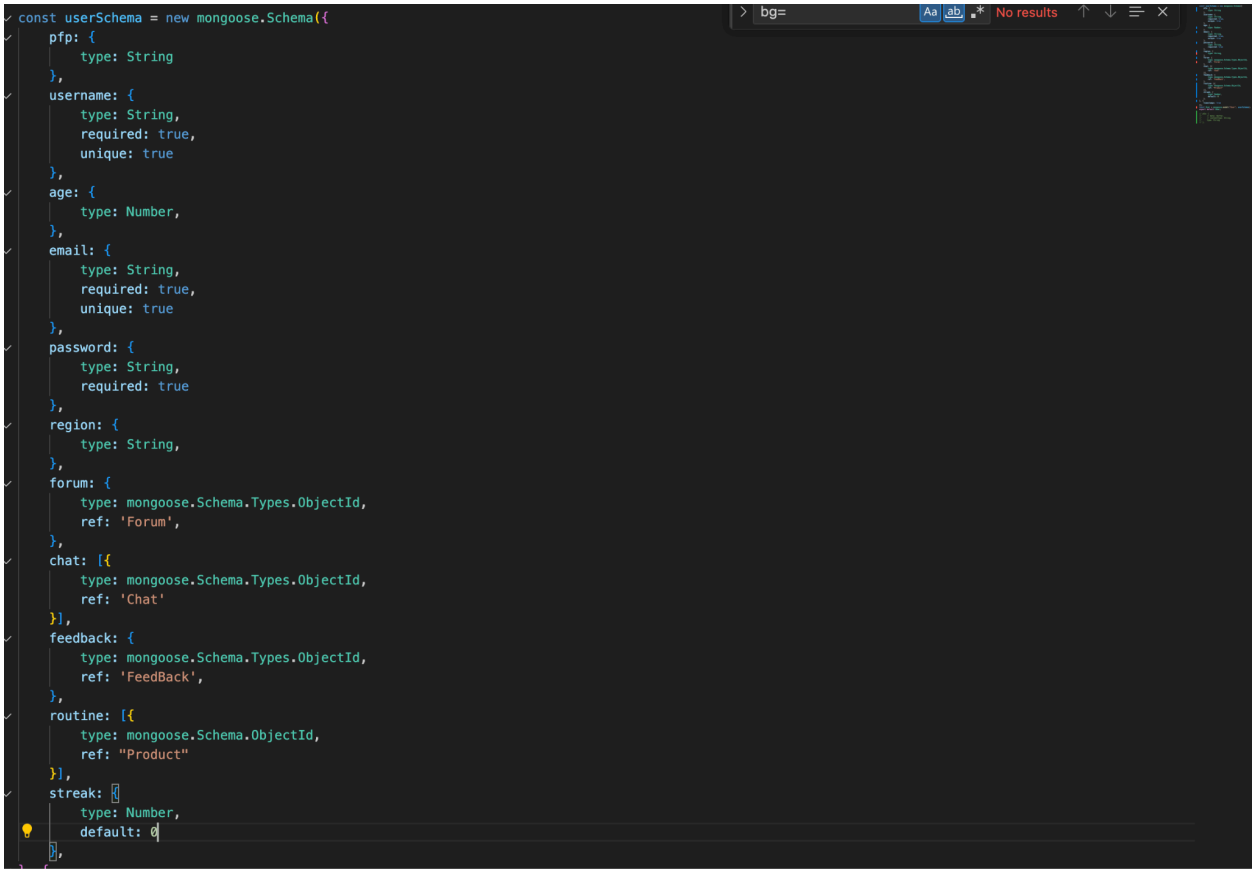


Fig 1: UML Data Diagram

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,
        unique: 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
    },
```

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 leaderboardSchema = new mongoose.Schema({
    users: [{
        type: mongoose.Schema.ObjectId,
        ref: "User"
    }],
    isCumulative: {
        type: Boolean,
    },
    isGlobal: {
        type: Boolean,
    }
}, {
    timestamps: true // Automatically manage createdAt and updatedAt
});

const Leaderboard = mongoose.model("Leaderboard", leaderboardSchema);

module.exports = Leaderboard;
```

Fig 4: Leaderboard Schema

Leaderboard schema contains a list of referenced users and two boolean type attributes being isCumulative and isGlobal. The two boolean values are for the filtering of the leaderboard. Depending on the filters, it will show different users within the leaderboard.

```
const forumSchema = new mongoose.Schema({
    answers: [{
        type: String,
        required: true
    }]
}, {
    timestamps: true
});

const Forum = mongoose.model("Forum", forumSchema);

export default Forum;
```

Fig 5: Forum Schema

Forum schema has an array of answers which would be the user's selected answers from the survey the user goes through while signing up.

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

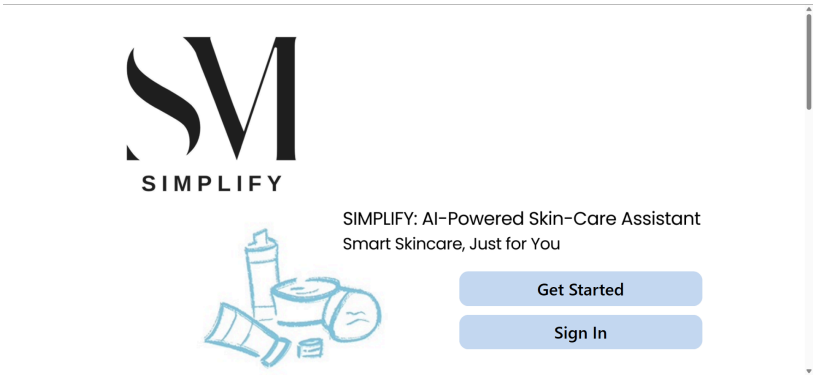
```
const chatSchema = new mongoose.Schema({
    query: {
        type: String,
        required: true
    },
    response: {
        type: String,
        required: false
    }
}, {
    timestamps: true // This will add createdAt and updatedAt fields automatically
});

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

export default Chat;
```

Fig 7: Chat Schema

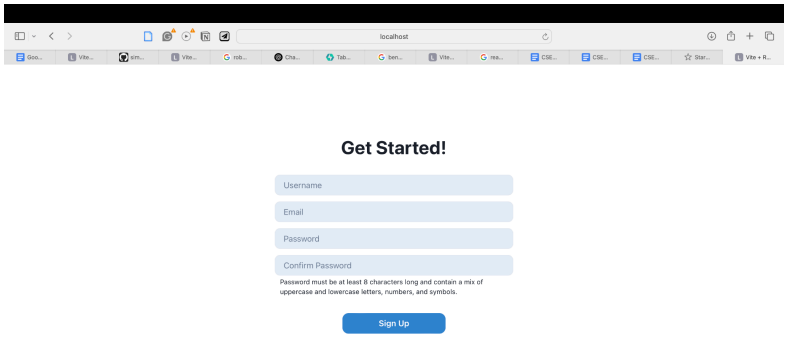
Chat schema is for storing each query and response for each user's query. By saving chats in an array, we will save the user chat history.

Web Views

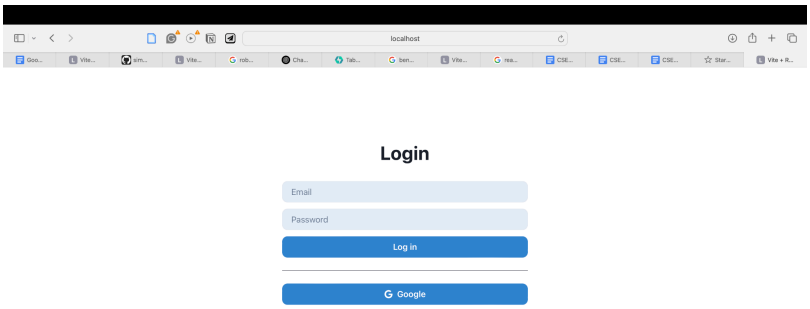
Welcome Page



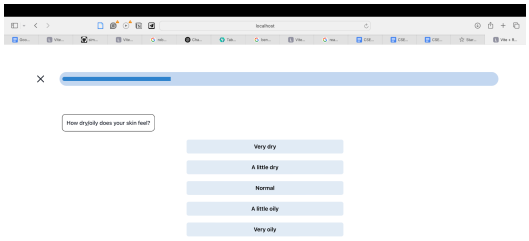
Sign Up Page

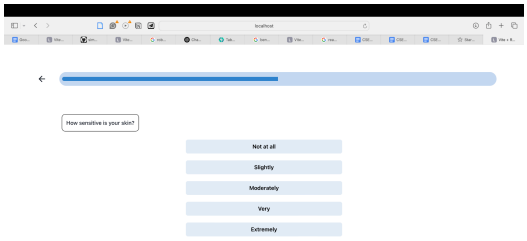


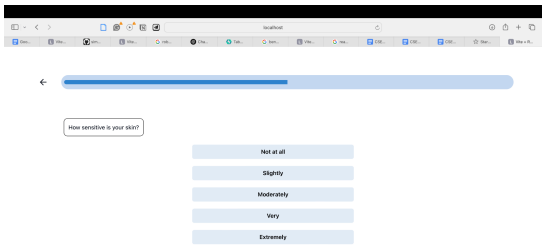
Login Page



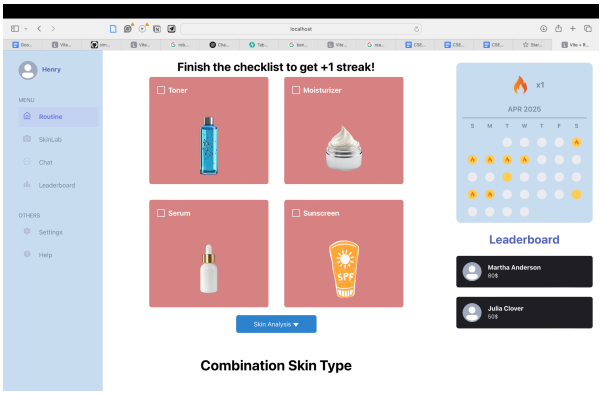
Survey Page

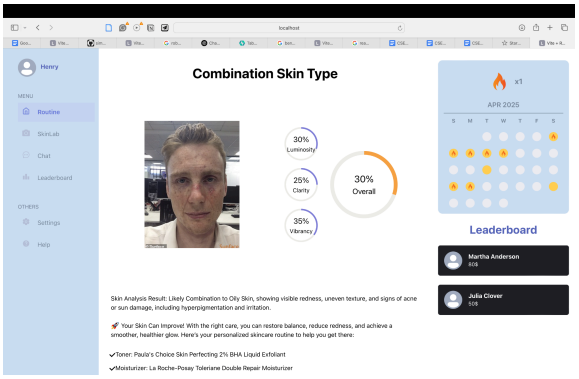




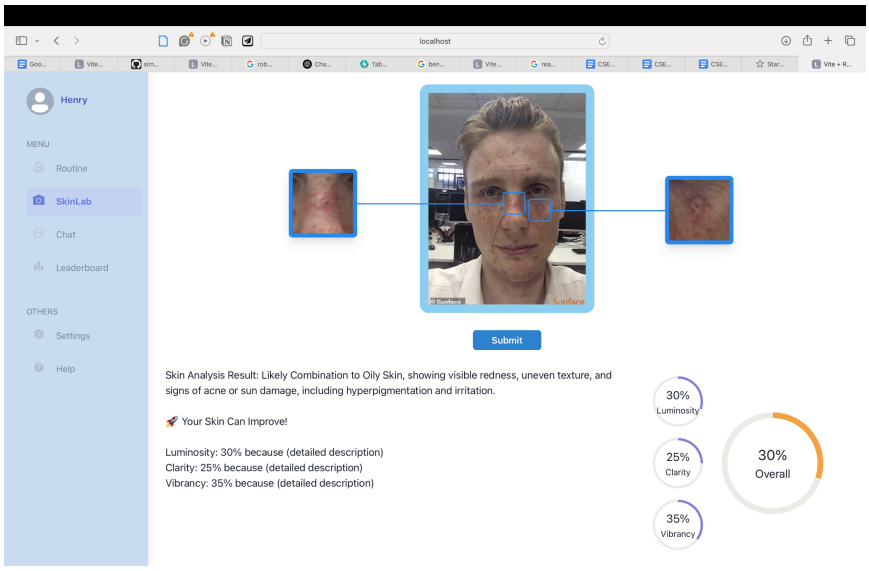


Home Page

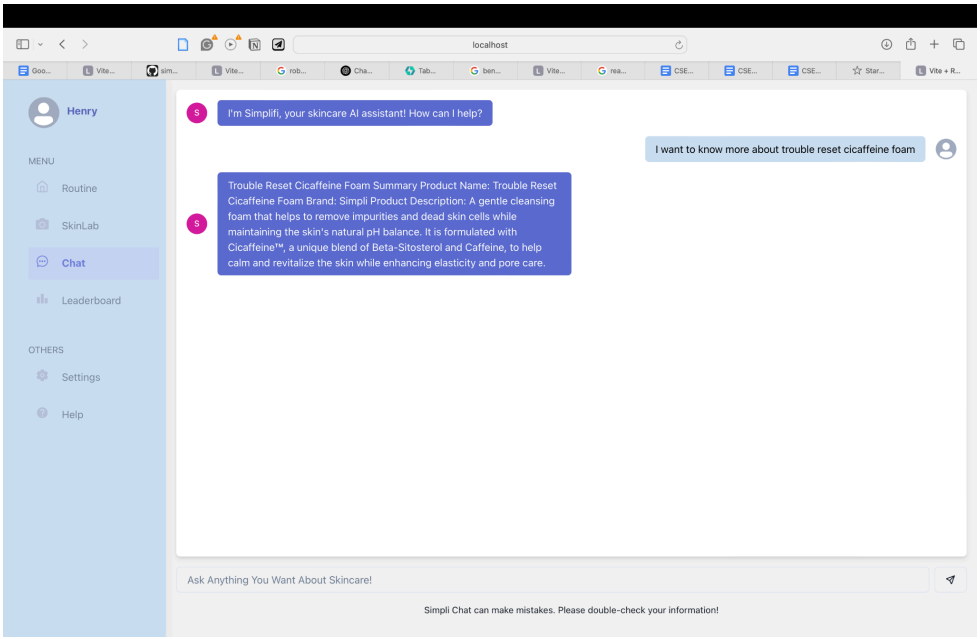




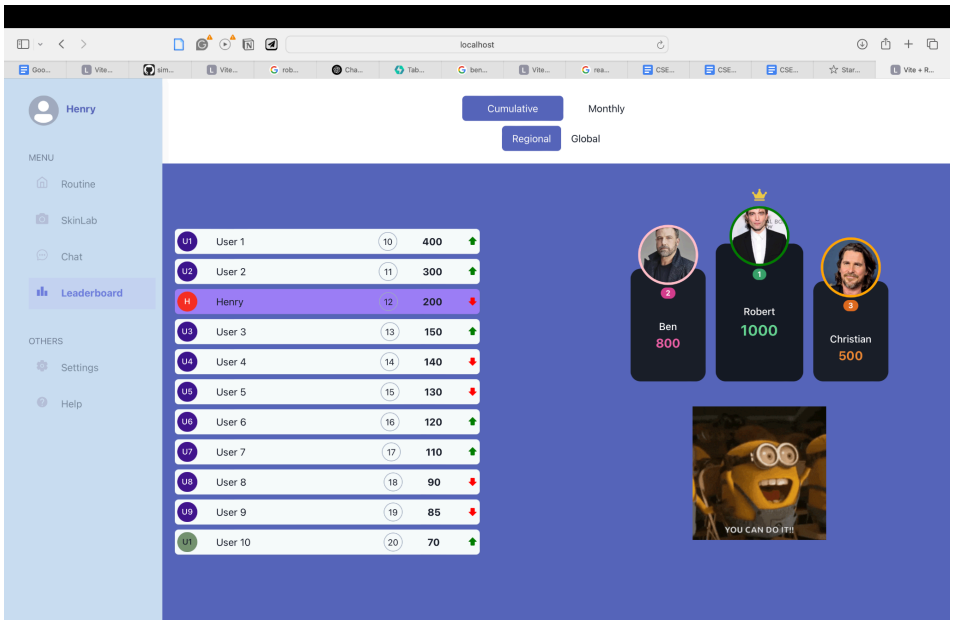
Skin Lab Page



Chat Page



Leaderboard Page



Contributions

1. Hein Zaw
    - MongoDB Schema design
    - LucidChart UML Diagram
    - ChatPage
    - Leaderboard Page
    - OpenAI backend
2. Minji Kim
    - Shema design
    - UML design
    - Homepage
    - Surveypage
3. Dowon Kim
    - Schema design
    - UML Class diagram
    - Welcome page
    - Signup + Login pages
    - Skin Lab page