AI-Driven Personalized Fitness and Nutrition Recommendation System

Team. MDN

TABLE OF CONTENTS

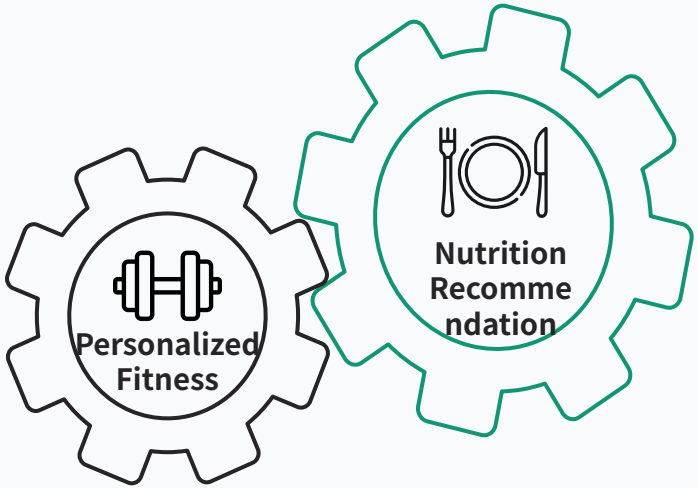
01

Project Overview

1. Item Name
2. Reason for Item Selection
3. Benchmarking Cases
4. Key Features of the Item
5. Unique Selling Points
6. Expected Benefits

01 Project Overview

1) Item Name



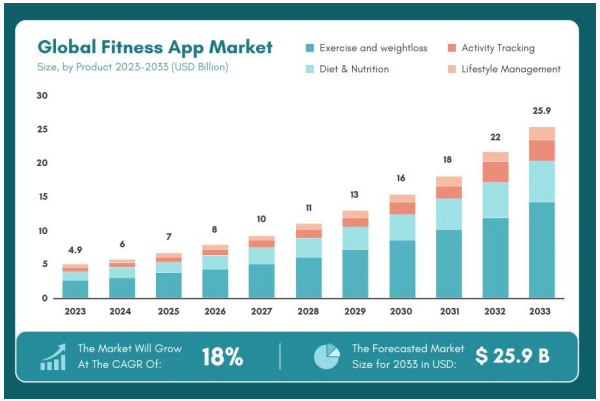
“AI-Driven Personalized Fitness and Nutrition Recommendation System”

HealthKitchen

01 Project Overview

2) Reason for Item Selection

- As public interest in health and wellness continues to grow, numerous fitness-related mobile applications have been released, and the overall industry has seen rapid growth.
- However, despite this growth, fitness apps tend to show low user retention rates. Although users are interested in health, they often fail to use fitness apps consistently over time.
- To address this issue, our team aimed to develop a solution that **motivates users to continuously engage with health apps**, thereby improving long-term usage and effectiveness.



source: FaviconNimble AppGenie - Fitness Industry Statistics: Trends and Insights

Mobile app retention benchmarks

To give you an idea of what mobile app retention rate benchmarks look like for different industries, here are examples of retention rates on day 30 following installations worldwide:

Mobile App Industry	Retention Rate on Day 30
News	11.3%
Business	5.1%
Shopping	5%
Finance	4.6%
Music	3.8%
Food & Drink	3.7%
Health & Fitness	3.7%
Lifestyle	3.6%
Productivity	3.2%
Video players	3.1%
Entertainment	3%
Travel	3%
Social	2.8%
Utilities	2.6%
Gaming	2.4%
Education	2.1%
Photography	1.5%

source: UXCam - Mobile App Retention Benchmarks

01 Project Overview

3) Benchmarking Cases -1

“Samsung Health”

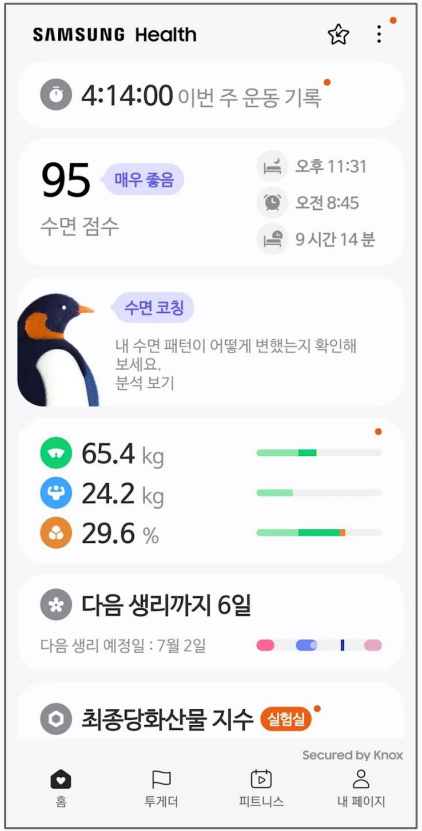
A health and fitness management app developed by Samsung Electronics.
It helps users manage their health by tracking and analyzing their physical activities, diet, sleep patterns, heart rate, and more.

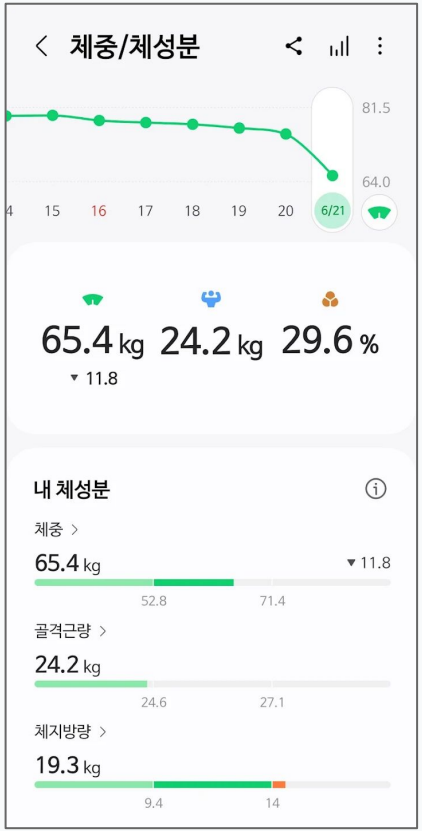
Strengths

Capable of tracking a wide range of health data
User-friendly UI
Seamless integration with Samsung devices

Weaknesses

Lacks diversity in workout types
Diet-related features are only for **logging purposes**, not for **guidance or recommendations** (Users must input detailed data such as calories themselves — without this, the app's effectiveness is limited)





01 Project Overview

3) Benchmarking Cases -2

“Fitness Online”

“Fitness Online” is an online fitness training app that offers professional workout programs and allows users to set up personalized fitness plans.

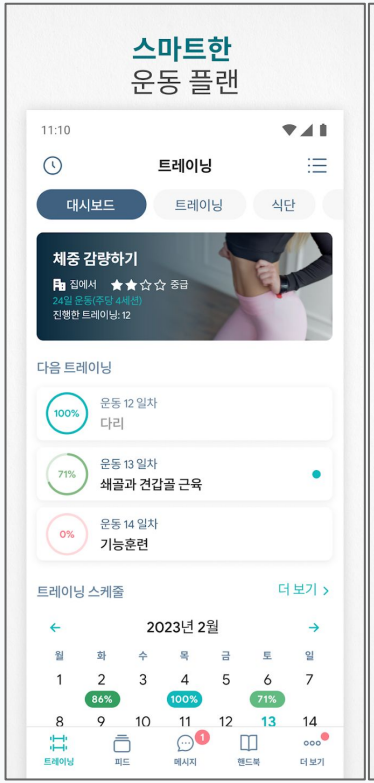
Strengths

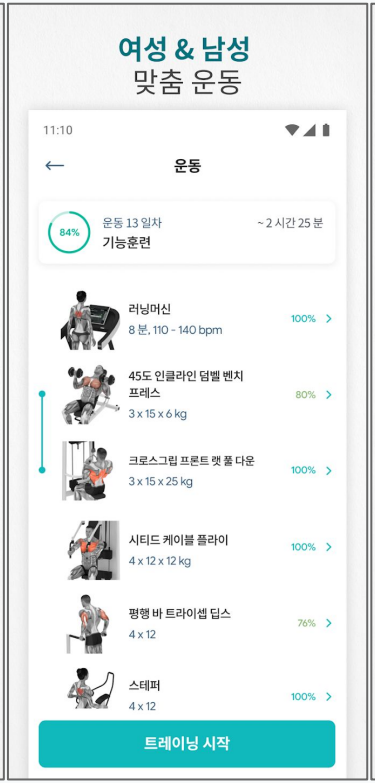
Offers online training features
Provides a variety of workout programs
Recommends workout plans tailored to individual goals

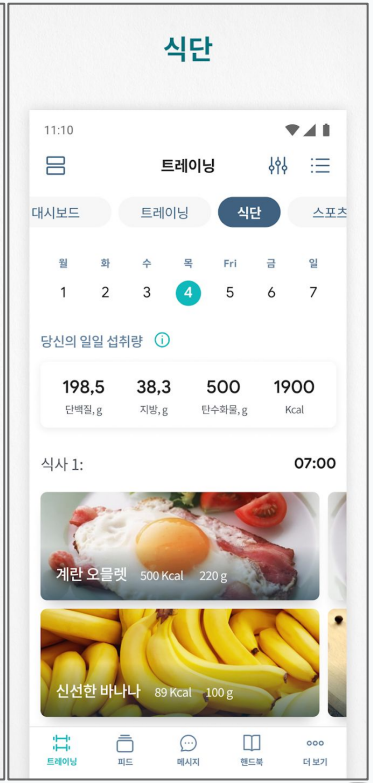
Weaknesses

For users who want a simple and lightweight fitness experience, the program content may feel overwhelming

The app only provides basic information on diet; users must **plan and execute their own meals manually**







01 Project Overview

4) Key Features of the Item

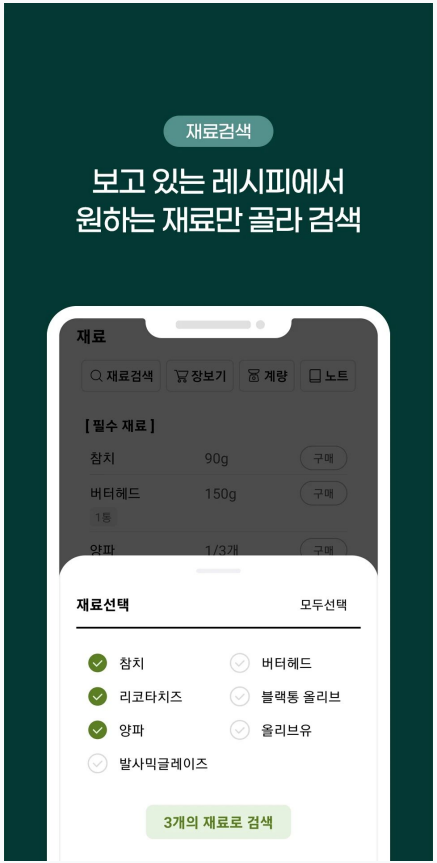
“Providing Meal Plans and Recipes Based on Fitness Goals”

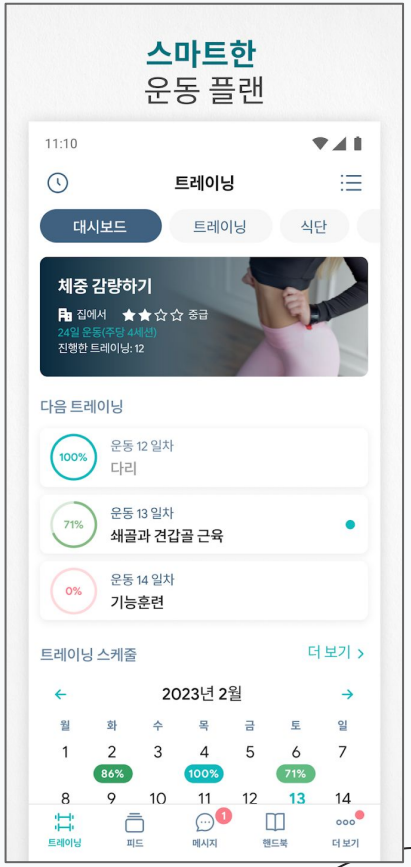
Developed an app that complements both the strengths and weaknesses of the benchmarked services

When a user sets their desired fitness goal, the AI recommends suitable exercises and workout methods

In addition to workouts, the app provides **appropriate meal plans** and corresponding **recipes**, recognizing diet as an equally important factor

Offers motivation features to help users continue using the app consistently over time



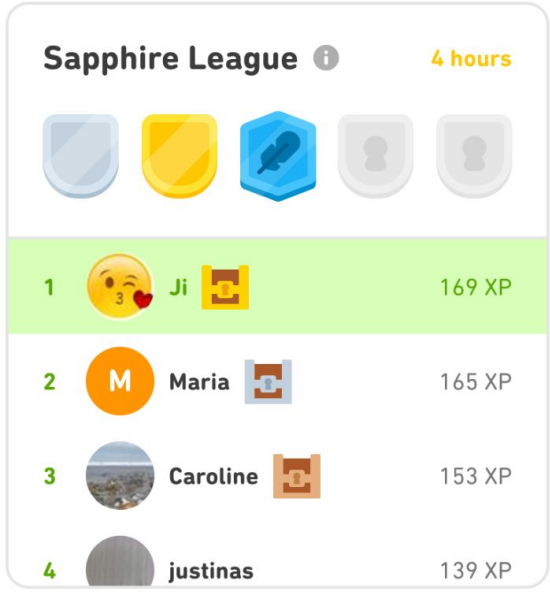


01 Project Overview

5) Unique Selling Points of the Item

"Benchmarking the Strategy of Duolingo, a Pioneer of the Daily Active User Model"

"Providing User Competition and Reward Systems"



<source: Duolingo>

01 Project Overview

6) Expected Benefits

1. Existing apps did not offer personalized meal plan recommendations or recipe suggestions, so our service provides users with the opportunity to manage their diet by cooking themselves adding novelty and differentiation to the experience.

2. Through **customized workout recommendations** based on time and space constraints, users can more effectively manage their personal health.

3. Features such as **notifications and reward systems** help users stay motivated and consistently manage their health, encouraging them to use the app frequently.

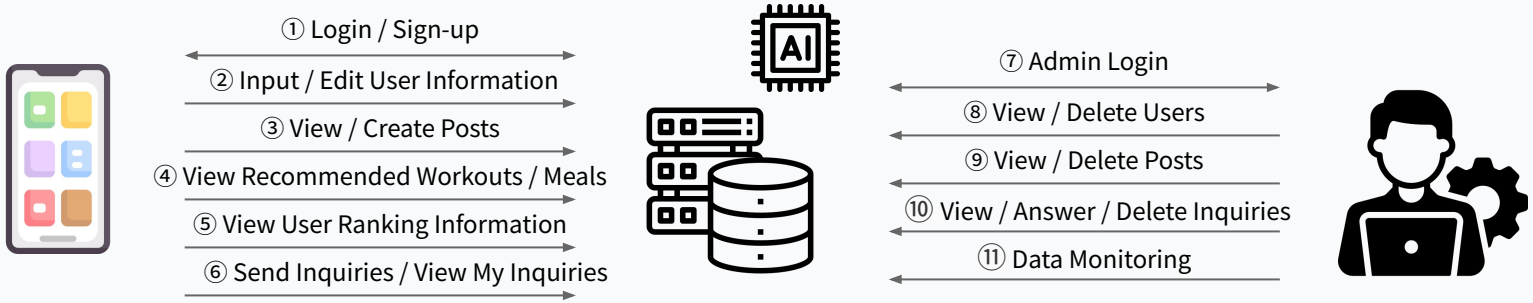
02

Project Development Details

1. Service Scenario
2. System Architecture
3. AI Model Development

02 Project Development Details

1) Service Scenario



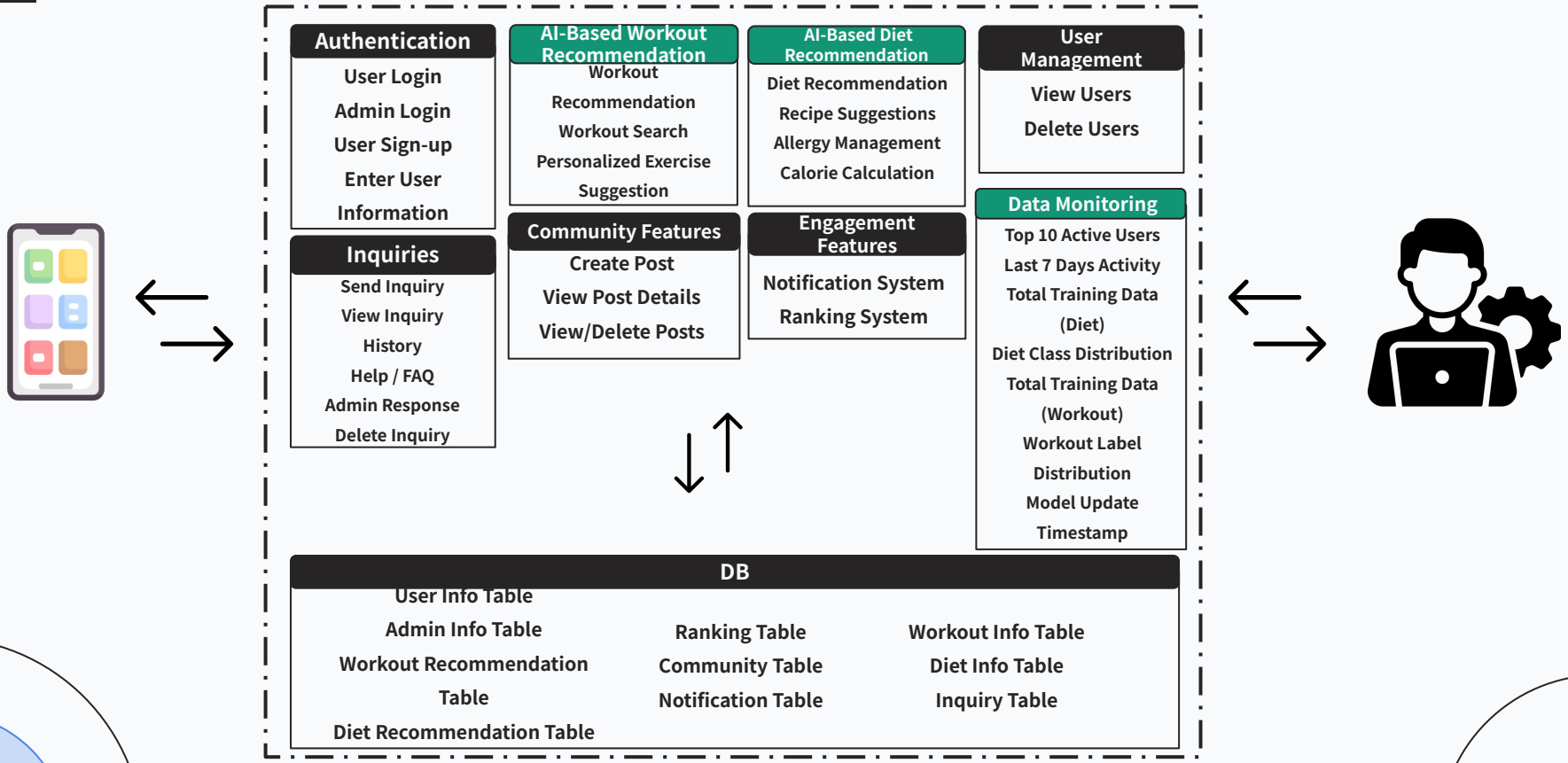
- Verifies user credentials upon login/sign-up
- Receives user profile, body info, workout and meal preferences
- Generates AI-based personalized workout and diet recommendations including recipes
- Provides workout and meal detail info on request
- Awards points when users complete workouts or meals
- Updates user points and logs accordingly
- Returns requested post content
- Displays user ranking based on recent activity
- Sends/receives inquiry messages

- The user requests ranking information.
- The server returns the ranking data.
- The user sends an inquiry email.
- The administrator sends a reply to the inquiry.
- The user requests to view their inquiry history.
- The server returns the inquiry content along with replies.
- The user edits their profile, body information, workout, and diet preferences.
- The server updates the user's information accordingly.

02

Project Development Details

2) System Architecture



02

Project Development Details

3) AI

AI Datasets	
Understanding User Body Information	UCI Obesity Dataset
Workout Recommendation Service	ExeciseDB API
	gym recommendation
Meal and Recipe Recommendation Service	Food/Nutrition Dataset

Data Preprocessing

We collect and structure key user features related to workout recommendation such as body measurements, BMI, activity level, and joint issues

↓

Labeling

Label training data based on user condition and workout goals to create structured, goal-specific learning targets

↓

Workout Recommendation

Utilize a multi-label classification model based on **CNN + SEBlock** to recommend multiple exercises simultaneously, based on the user's input features.

↓

Meal & Recipe Recommendation - 1

Vectorize nutritional and ingredient data from meals using TF-IDF based on the user's health status and preferences

↓

Meal & Recipe Recommendation - 2

Use **content-based filtering** (cosine similarity) to suggest appropriate meals and recipes for the user

↓

User Data Accumulation

As more user data is collected over time, we apply **collaborative filtering** to improve future recommendations (based on similar users)

↓

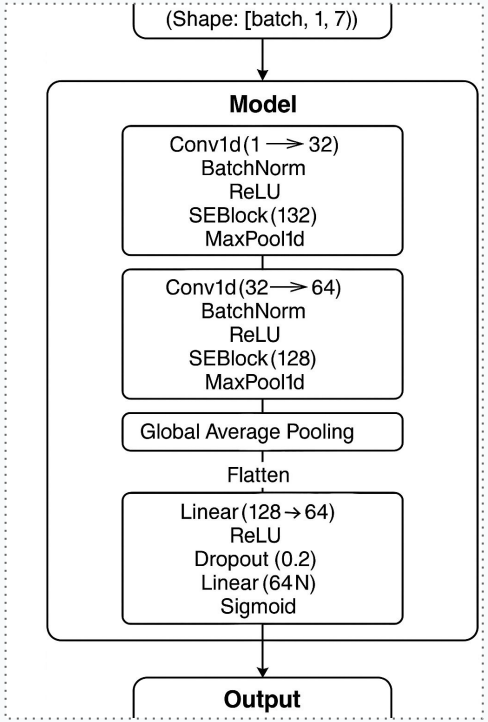
Personalized Workout Recommendation (Final Stage)

In the long term, we aim to provide **high-level personalized workout recommendations** using a hybrid recommendation algorithm that combines content-based and collaborative methods

02

Project Development Details

3) AI - Workout Recommendation



Workout Recommendation Model Based on CNN and SEBlock

1. **The model receives 7 user-specific features**, including physical information and workout goals, and recommends suitable exercises using a **multi-label classification** approach tailored to each individual.

2. The architecture combines **1D Convolutional Neural Networks (Conv1d)** with **Squeeze-and-Excitation (SE) Blocks** to effectively extract meaningful features from the input data.

3. The input passes through **three Conv1d + SEBlock layers**, with each layer increasing the number of channels. **MaxPooling** is used after each layer to summarize the extracted features.

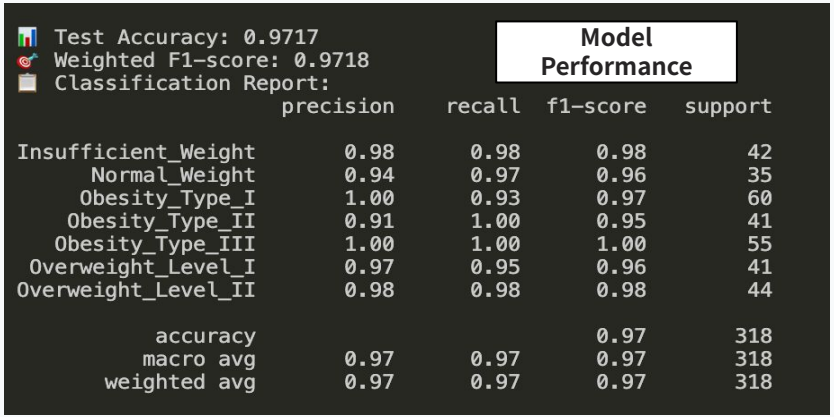
4. The **SEBlocks assign adaptive weights** to features, allowing the model to highlight fine-grained differences. This enables **highly personalized recommendations** and improved model performance **accuracy increased by approximately 3~4%.**

5. After passing through **Global Average Pooling**, the output is fed into an **MLP (Multi-Layer Perceptron)** classifier to calculate **scores (probabilities)** for each possible exercise label.

6. Finally, a **Sigmoid activation function** is applied to the output layer, allowing the model to recommend **multiple exercises simultaneously**, sorted by score.

02 Project Development Details

3) AI - obesity classification



Hyperparameter Settings

Parameter	Value / Setting	Description
Input Features	5	Features used as model inputs
Hidden Layers	2 layers (each with 64 neurons)	ReLU activation applied
Output Classes	7	7-level obesity classification (NObeyesdad labels)
Optimizer	Adam	Adaptive gradient-based optimizer
Learning Rate	0.001	Learning rate for optimizer
Loss Function	CrossEntropyLoss	Loss function for multi-class classification
Batch Size	32	Mini-batch size for training
Epochs	30	Number of total training iterations
Normalization	StandardScaler	Feature scaling method
Data Split	Train/Val/Test = 70%/15%/15%	Dataset split ratio

02 Project Development Details

3) AI – Workout Recommendation

Workout Recommendation Model Based on CNN and SEBlock

```python
# --- 모델 정의: CNN + SEBlock + MLP ---
class SEBlock(nn.Module):
    def __init__(self, channels, reduction):
        super().__init__()
        self.fc = nn.Sequential(
            nn.AdaptiveAvgPool1d(1),
            nn.Flatten(),
            nn.Linear(channels, channels//reduction),
            nn.ReLU(inplace=True),
            nn.Linear(channels//reduction, channels),
            nn.Sigmoid()
        )
    def forward(self, x):
        w = self.fc(x).unsqueeze(-1)
        return x * w

class CNNSEModel(nn.Module):
    def __init__(self, input_len, output_dim, base_ch, reduction, dropout):
        super().__init__()
        chs = [base_ch, base_ch*2, base_ch*4]
        layers, in_ch = [], 1
        for c in chs:
            layers += [
                nn.Conv1d(in_ch, c, kernel_size=3, padding=1),
                nn.BatchNorm1d(c), nn.ReLU(inplace=True),
                SEBlock(c, reduction),
                nn.MaxPool1d(2,2,ceil_mode=True)
            ]
            in_ch = c
        self.features    = nn.Sequential(*layers)
        self.global_pool = nn.AdaptiveAvgPool1d(1)
        self.classifier  = nn.Sequential(
            nn.Flatten(),
            nn.Linear(chs[-1], chs[-1]//2),
            nn.ReLU(inplace=True),
            nn.Dropout(dropout),
            nn.Linear(chs[-1]//2, output_dim),
            nn.Sigmoid()
        )
    def forward(self, x):
        x = self.features(x)
        x = self.global_pool(x)
        return self.classifier(x)
```

하이퍼파라미터	값	설명
Learning Rate	0.001	AdamW Optimizer의 학습률
Weight Decay	0.00001	AdamW 옵티마이저의 정규화 계수
Dropout	0.2	MLP 분류기 중간의 드롭아웃 비율
SEBlock Reduction	8	SEBlock 내부 채널 축소 비율
Base Channel	32	Conv1d 첫 출력 채널 수
Batch Size	32	한 번에 처리하는 데이터 샘플 수
Epochs	30	전체 학습 반복 횟수
Threshold	0.43	추천 확률을 1/0으로 나누는 기준
Optimizer	AdamW	최적화 알고리즘
Loss	BCELoss	이진 교차 엔트로피 (멀티레이블용)
Test Ratio	0.2	테스트셋 비율
Random State	42	데이터셋 분할 시 랜덤 시드

```
📊 최종 테스트 결과 (threshold 최적화 적용):
Threshold:          0.43
Precision (micro):  0.7729
Recall    (micro):  0.9735
F1-score  (micro):  0.8617
Hamming Loss:       0.1142
```

02

Project Development Details

3) AI – Workout Recommendation

Hyperparameter Tuning

```
[2/128] 조합 : {'lr': 0.001, 'batch_size': 16, 'dropout': 0.1, 'base_ch': 16, 'reduction': 4, 'weight_decay': 0, 'threshold': 0.43}
F1: 0.8582, Hamming Loss: 0.1151

[3/128] 조합 : {'lr': 0.001, 'batch_size': 16, 'dropout': 0.1, 'base_ch': 16, 'reduction': 4, 'weight_decay': 1e-05, 'threshold': 0.4}
F1: 0.8620, Hamming Loss: 0.1149

[4/128] 조합 : {'lr': 0.001, 'batch_size': 16, 'dropout': 0.1, 'base_ch': 16, 'reduction': 4, 'weight_decay': 1e-05, 'threshold': 0.43}
F1: 0.8544, Hamming Loss: 0.1149

[5/128] 조합 : {'lr': 0.001, 'batch_size': 16, 'dropout': 0.1, 'base_ch': 16, 'reduction': 8, 'weight_decay': 0, 'threshold': 0.4}
F1: 0.8609, Hamming Loss: 0.1152

[6/128] 조합 : {'lr': 0.001, 'batch_size': 16, 'dropout': 0.1, 'base_ch': 16, 'reduction': 8, 'weight_decay': 0, 'threshold': 0.43}
F1: 0.8557, Hamming Loss: 0.1162
```

- In this project, we optimized the model's performance by systematically adjusting multiple hyperparameters through a **Grid Search process**.
- We first selected key hyperparameters known to significantly impact performance — including **learning rate, batch size, dropout rate, number of convolutional channels, SEBlock reduction ratio, weight decay**, and **prediction threshold**.
- We defined a set of candidate values for each hyperparameter, and tested all possible combinations using the **Grid Search method**.
- For each combination, the model was trained using the same dataset, and evaluation metrics such as **F1-score** and **Hamming Loss** were calculated to measure how well the model could recommend workouts.
- Initial experiments were conducted with **5-epoch short runs** to quickly filter out low-performing combinations. The most promising candidates were then trained further to obtain the final model.
- Based on these experiments, we selected the best-performing combination of hyperparameters, and successfully built the **optimized CNN + SEBlock model**.

Diet Recommendation Flow

User Data Input
↓
BMI Calculation
↓
Data Preprocessing
↓
BaseMeal Prediction
(via PyTorch Neural Network)
↓
Meal Selection Based on Fitness Goal
→ *Lite (Low-Calorie)* or *XL (High-Calorie)*
↓
Return Detailed Information
(from meal_variants.csv: calories, ingredients, recipe)

1. The user's height and weight are used to calculate BMI, which is then used to predict the most appropriate **BaseMeal**.
2. The predicted meal is adjusted based on the user's fitness goal: either a **low-calorie (Lite)** or **high-calorie (XL)** version. The system provides not only calorie values, but also **ingredient lists and cooking instructions**.
3. To avoid overfitting, instead of using a complex model, we built a **lightweight 3-layer neural network** that delivers **fast and stable meal recommendations**.

02 Project Development Details

3) AI – Diet Recommendation

Model Definition

```python
class MealNet(nn.Module):
    def __init__(self, in_dim: int, out_dim: int):
        super().__init__()
        self.net = nn.Sequential(
            nn.Linear(in_dim, 128),        # 입력 → 128개 뉴런
            nn.BatchNorm1d(128),           # 배치 정규화: 학습 안정화 및 가속
            nn.ReLU(),                     # 비선형 활성화 함수
            nn.Dropout(0.3),               # 과적합 방지 (30% 무작위 드롭)
            nn.Linear(128, 64),            # 128 → 64개 뉴런
            nn.ReLU(),                     # 다시 활성화 함수
            nn.Linear(64, out_dim)         # 최종 출력: 식단 클래스 개수
        )
    def forward(self, x): return self.net(x)

device    = torch.device("cpu")
model     = MealNet(X.shape[1], len(meal_le.classes_)).to(device)
```

Model Performance

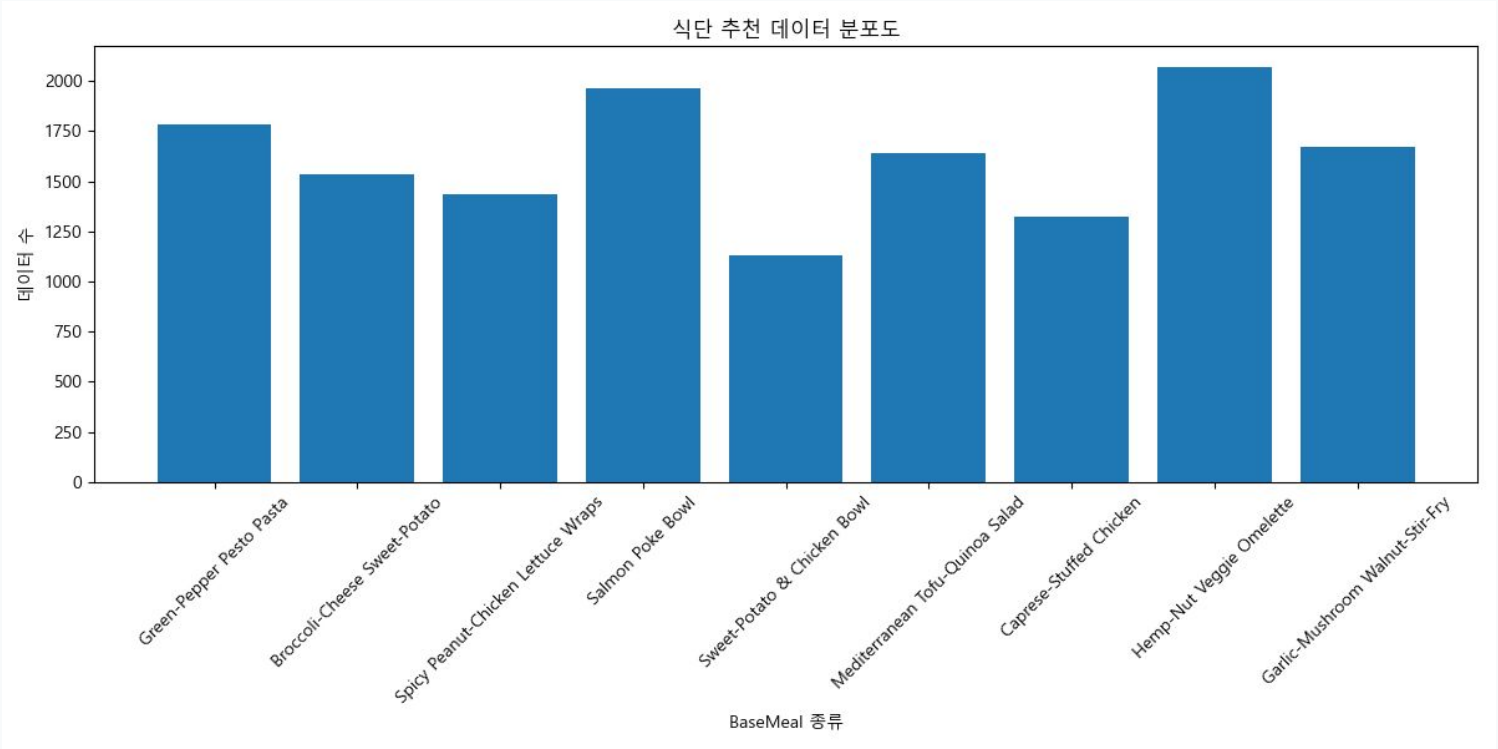
Classification Report:

	precision	recall	f1-score
Broccoli-Cheese Sweet-Potato	0.95	0.94	0.94
Caprese-Stuffed Chicken	0.94	0.89	0.91
Garlic-Mushroom Walnut Stir-Fry	0.88	0.91	0.89
Green-Pepper Pesto Pasta	0.94	0.96	0.95
Hemp-Nut Veggie Omelette	0.90	0.88	0.89
Mediterranean Tofu-Quinoa Salad	0.97	0.90	0.93
Salmon Poke Bowl	0.93	0.94	0.94
Spicy Peanut-Chicken Lettuce Wraps	0.93	0.90	0.91
Sweet-Potato & Chicken Bowl	0.89	0.93	0.91
accuracy			0.93
macro avg	0.92	0.92	0.92
weighted avg	0.93	0.93	0.93

Project Development Details

3) AI – Diet Recommendation dataset

Diet Recommendation Data Distribution



Diet Recommendation Algorithm

```python
def recommend_meal(user: dict, top_k: int = 3, temperature: float = 5.0, sim_weight=0.3):

    # 3-a. 입력값 보완
    user = user.copy()
    bmi = round(user["Weight"] / user["Height"]**2, 2)
    user["BMI"] = bmi
    user["Level"] = bmi_to_level(bmi)

    # 3-b. 전처리
    num_cols = ["Age", "Height", "Weight", "BMI"]
    cat_cols = ["Sex", "Hypertension", "Diabetes", "Level", "Fitness Goal"]

    X_num = scaler.transform([[user[c] for c in num_cols]])
    X_cat = np.column_stack([encoders[c].transform([user[c]]) for c in cat_cols])
    X = np.hstack([X_num, X_cat]).astype(np.float32)
    X_t = torch.tensor(X)

    # 3-c. 예측 + 코사인 유사도 결합
    with torch.no_grad():
        # 모델 예측
        logits = model(X_t) / temperature
        model_probs = torch.softmax(logits, dim=1).numpy()[0]

        # 코사인 유사도 계산
        user_vec = X[0]
        sim_scores = []
        for cls in range(len(meal_le.classes_)):
            meal_vec = meal_embeddings[cls]
            # 코사인 유사도 공식 적용
            dot_product = np.dot(user_vec, meal_vec)
            norm_user = np.linalg.norm(user_vec)
            norm_meal = np.linalg.norm(meal_vec)
            sim = dot_product / (norm_user * norm_meal + 1e-8)
            sim_scores.append(sim)

        # 최종 점수 계산 (모델 70% + 유사도 30%)
        final_scores = (1-sim_weight)*model_probs + sim_weight*np.array(sim_scores)
```

```python
    # 3-d. 결과 선정
    k = min(top_k, len(final_scores))
    top_idx = final_scores.argsort()[-k:][::-1]
    top_p = final_scores[top_idx]

    # 3-e. 결과 구성
    recs = []
    for idx, p in zip(top_idx, top_p):
        base_meal = meal_le.inverse_transform([idx])[0]
        size = "XL" if user["Fitness Goal"] == "Weight Gain" else "Lite"
        meta = variants_dict.get((base_meal, size))

        rec = {
            "Meal": f"{base_meal} [{size}]",
            "Prob": float(p),
            "Calories": "N/A",
            "Ingredients": "등록되지 않음",
            "Instructions": "등록되지 않음"
        }
        if meta:
            rec.update({
                "Calories": meta["Calories"],
                "Ingredients": meta["Ingredients"],
                "Instructions": meta["Instructions"]
            })
        recs.append(rec)
    return recs
```

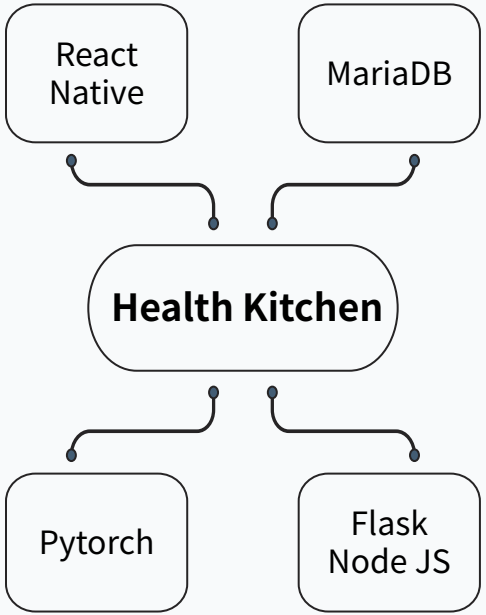
03

Project Development Strategy

1. Team Structure and Roles
2. Development Schedule and Plan

03 Project Development Strategy

1) Team Structure and Roles



FE	이세영	1) UI/UX design and layout 2) API integration and data visualization 3) Implementation of notification and social features
BE	김민규 정상윤	1) Server development and management 2) API development and data handling 3) User authentication and security 4) Integration with AI models
AI	오흥찬 김경환	1) Development of personalized workout and diet recommendation AI 2) Data collection and model performance optimization 3) Implementation of AI feedback system

03 Project Development Strategy

2) Development Schedule and Plan

To Do		week1	week2	week3	week4	week5	week6	week7	week8	week9	week 10	week 11	week 12	week 13	week 14	week 1
Analysis & Design	Proposal	Requirement Analysis														
	Design Document			Design Phase												
Implementation & Testing	Programming					Implementation Phase										
	Midterm Evaluation													Deliverables		
Finalization	Final Report															
	Final Presentation															

04

Final Outcomes

1. Source Code
2. Project Demonstration

04 Final Outcomes

1) Source Code – Mobile App (Frontend)

```
{/* 추천 운동 */}
<Text style={styles.sectionTitle}>{userName}님의 맞춤 운동</Text>
<ScrollView horizontal showsHorizontalScrollIndicator={false} style={styles.horizontalScroll}>
  {exerciseList.map((name, index) => {
    const key = name.toLowerCase();  // 운동 이름 소문자 변환
    const image = workoutImages[key] || workoutImages["default"];

    return (
      <TouchableOpacity
        key={index}
        onPress={() => {
          router.push({ pathname: '/workout-detail', params: { name } });
        }}
        style={{ alignItems: 'center', marginRight: 8 }}  // ✅ 텍스트 정렬용
      >
        <View style={styles.workoutCard}>
          <Image source={image} style={styles.thumbnail} />
        </View>
        <Text style={styles.cardText}>{name}</Text>  {/* ✅ 박스 바깥으로 나감 */}
      </TouchableOpacity>
    );
  })}
</ScrollView>
```

User Home Screen: AI-Based Personalized Workout Recommendations

```
{/* 추천 식단 */}
<Text style={styles.sectionTitle}>{userName}님의 맞춤 식단</Text>
<ScrollView horizontal showsHorizontalScrollIndicator={false} style={styles.horizontalScroll}>
  {dietList.map((item, index) => {
    const image = getDietImage(item);
    return (
      <TouchableOpacity
        key={index}
        onPress={async () => {
          await AsyncStorage.setItem('selected_diet', item);
          router.push('/diet-detail');
        }}
        style={{ alignItems: 'center', marginRight: 8 }}
      >
        <View style={styles.workoutCard}>
          <Image source={image} style={styles.thumbnail} />
        </View>
        <Text style={styles.cardText}>{item}</Text>
      </TouchableOpacity>
    );
  })}
</ScrollView>
```

User Home Screen: AI-Based Personalized Meal/Recipe Recommendations

04 Final Outcomes

1) Source Code – Mobile App Backend (Diet)

```python
def recommend_diet_by_ai(user: dict) -> dict:
    # BMI 계산 및 체형 등급 추가
    bmi, level = bmi_and_level(user["Weight"], user["Height"])
    user["BMI"] = bmi
    user["Level"] = level

    # 범주형 인코딩
    for c in CAT_COLS:
        le = encoders[c]
        user[c] = le.transform([user[c]])[0] if user[c] in le.classes_ else 0

    # 입력 벡터 구성
    x_num = scaler.transform([[user[c] for c in NUM_COLS]])
    x_cat = np.array([[user[c] for c in CAT_COLS]], dtype=float)
    x     = np.hstack([x_num, x_cat])

    # 예측 및 결과 반환
    with torch.no_grad():
        logits = model(torch.tensor(x, dtype=torch.float32))
        probs  = torch.softmax(logits, dim=1).numpy()[0]
```

Recommendation Model Invocation and Preprocessing Logic

```python
@diet_bp.route('/flask/diet/recommend/ai', methods=['POST'])
def diet_ai_recommend():
    try:
        user = request.get_json()
        recommended = recommend_diet_by_ai(user)
        return jsonify({"status": "success", "data": recommended, "error": None}), 200
    except Exception as e:
        return jsonify({"status": "error", "data": None, "error": str(e)}), 500
```

API Routing and Integration or **API Endpoint Routing**

04 Final Outcomes

1) Source Code – Mobile App Backend (Fitness)

```python
def get_exercise_recommendation(user_id, sex, age, height, weight,
                                hypertension, goal, level, has_membership):
    # 사용자 입력값 전처리 (단위 변환 및 구조화)
    # 프론트에서 받은 사용자 데이터를 모델이 처리할 수 있는 형식으로 가공
    user_input = {
        "Sex": sex,
        "Age": int(age),
        "Height": float(height) / 100,   # cm → m 변환
        "Weight": float(weight),
        "Hypertension": hypertension,
        "Fitness Goal": goal,
        "Level": level,
        "Has Gym Membership": has_membership
    }

    # AI 모델 호출 (🧠 AI 팀원이 사전 학습하여 제공한 CNN+SEBlock 모델)
    # - 학습된 모델을 재학습하지않는 방식으로 호출
    recommended = recommend_exercises_from_model(user_input)

    # 추천 결과 전처리
    # 쉼표 구분 문자열 형태로 변환하여 DB에 저장하기 위한 가공 처리
    cleaned = [ex.strip().lower() for ex in recommended]
    cleaned_str = ','.join(cleaned)
```

```python
    # 추천 결과를 exercise_logs 테이블에 저장
    # 결과를 조회하거나 랭킹 시스템에 활용할 수 있도록 기록
    conn = get_connection()
    cursor = conn.cursor()
    try:
        query = """
            INSERT INTO exercise_logs (uid, recommended_exercises, timestamp)
            VALUES (%s, %s, %s)
        """
        cursor.execute(query, (user_id, cleaned_str, datetime.now()))
        conn.commit()
    finally:
        cursor.close()
        conn.close()

    # 추천된 운동 리스트를 반환 (프론트와 연결되어 사용자에게 바로 표시됨)
    return recommended
```

Recommendation Model Invocation and Preprocessing Logic

04 Final Outcomes

1) Source Code – Data Preprocessing for Collaborative Filtering

```
        if row:
            # 📌 가중치는 랭킹순으로 부여
            rank = top_users.index(user) + 1
            weight = round(1.0 - (rank - 1) * 0.1, 2)

            # DB 테이블에 가중치 저장
            cursor.execute("""
                INSERT INTO exercise_logs (uid, exercises, weight)
                VALUES (%s, %s, %s)
                ON DUPLICATE KEY UPDATE exercises = VALUES(exercises), weight = VALUES(weight)
            """, (user['uid'], row['recommended_exercises'], weight))

            retrain_data.append({
                "uid": user['uid'],
                "nickname": user['nickname'],
                "recommended_exercises": row['recommended_exercises'],
                "weight": weight
            })

    conn.commit()
    return jsonify({"status": "success", "data": retrain_data}), 200
except Exception as e:
    return jsonify({"status": "error", "error": str(e)}), 500
finally:
    cursor.close()
    conn.close()
```

```
@exercise_bp.route('/flask/retrain-data', methods=['GET'])
def get_retrain_dataset():
    try:
        conn = get_connection()
        cursor = conn.cursor(dictionary=True)

        # 최근 7일 랭킹 기준 상위 10명 호출
        cursor.execute("""
            SELECT u.uid, u.nickname
            FROM users u
            JOIN exercise_logs e ON u.uid = e.uid
            WHERE e.recommended_at >= NOW() - INTERVAL 7 DAY
            GROUP BY u.uid
            ORDER BY SUM(e.score) DESC
            LIMIT 10
        """)
        top_users = cursor.fetchall()

        retrain_data = []
        for user in top_users:
            cursor.execute("""
                SELECT recommended_exercises
                FROM exercise_logs
                WHERE uid = %s
                ORDER BY timestamp DESC
                LIMIT 1
            """, (user['uid'],))
            row = cursor.fetchone()
```

Training Data Collection with Ranking-Based Weighting

04 Final Outcomes

1) Source Code – Admin Web Frontend (Data Monitoring)

Daily Log Count

```jsx
<div className="chart-card" style={{ flex: 2 }}>
    <h5>일간 로그 수 (그래프)</h5>
    {monitorData ? (
        <ResponsiveContainer width="100%" height={250}>
            <LineChart data={monitorData.recent_7days}>
                <CartesianGrid strokeDasharray="3 3" />
                <XAxis dataKey="date" />
                <YAxis />
                <Tooltip />
                <Line type="monotone" dataKey="count" stroke="#8884d8" />
            </LineChart>
        </ResponsiveContainer>
    ) : (
        <p>그래프 로딩 중...</p>
    )}
</div>
```

Distribution of Diet Classes

```jsx
{/* AI 학습 데이터 분포 */}
        <div className="chart-row">
            <div className="chart-card large">
                <h5>식단 클래스 분포</h5>
                {monitorData && (
                    <ResponsiveContainer width="100%" height={400}>
                        <PieChart>
                            <Pie
                                data={summarizeDistribution(monitorData.ai_data.diet_class_distribution, 10)}
                                dataKey="value"
                                nameKey="name"
                                outerRadius={150}
                                label
                            >
                                {summarizeDistribution(monitorData.ai_data.diet_class_distribution, 10).map((entry, index) => (
                                    <Cell key={`cell-${index}`} fill={COLORS[index % COLORS.length]} />
                                ))}
                            </Pie>
                            <Tooltip content={<CustomPieTooltip />} />
                        </PieChart>
                    </ResponsiveContainer>
                )}
            </div>
```

Top 10 Users by Activity

```jsx
<div className="chart-row">
    <div className="chart-card large">
        <h5>Top 10 사용자 활동량 (그래프)</h5>
        {monitorData ? (
            <ResponsiveContainer width="100%" height={600}>
                <BarChart
                    data={monitorData.top_users}
                    margin={{ top: 40, right: 30, left: 20, bottom: 160 }}
                >
                    <CartesianGrid strokeDasharray="3 3" />
                    <XAxis dataKey="uid" tick={{ angle: -45, textAnchor: 'end' }} interval={0} />
                    <YAxis type="number" domain={[0, 'dataMax']} allowDecimals={false} />
                    <Tooltip />
                    <Legend verticalAlign='top' align='center' wrapperStyle={{ paddingBottom: 30}} />
                    <Bar dataKey="total_logs" fill="#82ca9d" name="총 로그 수" barSize={30}  />
                </BarChart>
            </ResponsiveContainer>
        ) : (
            <p className='chart-text'>그래프 로딩 중...</p>
        )}
    </div>
</div>
```

Distribution of Exercise Labels

```jsx
<div className="chart-card large">
    <h5>운동 라벨 분포</h5>
    {monitorData &&
        monitorData.ai_data &&
        Object.keys(monitorData.ai_data.exercise_label_distribution).length > 0 ? (
        <ResponsiveContainer width="100%" height={400}>
            <BarChart
                data={Object.entries(monitorData.ai_data.exercise_label_distribution).map(
                    ([name, value]) => ({ name, value })
                )}
                margin={{ top: 20, right: 30, left: 20, bottom: 60 }}
            >
                <CartesianGrid strokeDasharray="3 3" />
                <XAxis
                    dataKey="name"
                    interval={0}
                    tick={{ angle: -30, textAnchor: 'end', fontSize: 12 }}
                    fontSize={12}
                />
                <YAxis allowDecimals={false} fontSize={12} />
                <Tooltip />
                <Legend verticalAlign="top" align="center" wrapperStyle={{ paddingBottom: 30 }}/>
                <Bar
                    dataKey="value"
                    fill="#8884d8"
                    name="운동 항목 수"
                    barSize={15}
                    label={{ position: 'top', fontSize: 12 }}
                />
            </BarChart>
        </ResponsiveContainer>
    ) : (
        <p className="chart-text">운동 라벨 데이터가 없습니다.</p>
    )}
</div>
```

04 Final Outcomes

1) Source Code – Admin Web Backend (AI Data & Model Status Monitoring)

```
// GET /admin/ai-monitor
// 관리자 페이지에서 전체 로그 수, 최근 7일 활동량, 상위 사용자, AI 학습 데이터 현황 및 모델 상태를 반환
router.get('/ai-monitor', verifyAdmin, async (req, res) => {
  try {
    // 1. 로그 개수
    const [[exerciseCount]] = await db.query('SELECT COUNT(*) AS count FROM exercise_logs');
    const [[dietCount]] = await db.query('SELECT COUNT(*) AS count FROM diet_logs');

    // 2. 최근 7일간 기록 (운동 + 식단)
    const [dailyLogs] = await db.query(`
      SELECT date, COUNT(*) AS count FROM (
        SELECT DATE(performed_at) AS date FROM exercise_logs
        UNION ALL
        SELECT DATE(consumed_at) AS date FROM diet_logs
      ) AS all_logs
      WHERE date >= DATE_SUB(CURDATE(), INTERVAL 7 DAY)
      GROUP BY date
      ORDER BY date ASC
    `);
```

```
// 3. 상위 활동 사용자 TOP10
const [userData] = await db.query(`
  SELECT uid,
    (SELECT COUNT(*) FROM exercise_logs e WHERE e.uid COLLATE utf8mb4_general_ci = u.uid COLLATE utf8mb4_general_ci) +
    (SELECT COUNT(*) FROM diet_logs d WHERE d.uid COLLATE utf8mb4_general_ci = u.uid COLLATE utf8mb4_general_ci) AS total_logs
  FROM users u
  ORDER BY total_logs DESC
  LIMIT 10
`);

// 4. 학습 데이터 통계 및 모델 수정일
const dietCSV = path.join(__dirname, '../../../ai/data/diet_recommend.csv');
const exerciseCSV = path.join(__dirname, '../../../ai/gym_for_recommendation.csv');
const dietModel = path.join(__dirname, '../../../models/diet_multilabel_net.pt'); // 저장된 모델명에 맞게 수정
const exModel = path.join(__dirname, '../../../models/multilabel_exercise_model.pt');

const dietStats = await getCSVStats(dietCSV, 'Meal');
const exStats = await getCSVStats(exerciseCSV, 'Exercises');

const dietModelTime = getModelModifiedTime(dietModel);
const exModelTime = getModelModifiedTime(exModel);
```

Total Log Count (Workout + Diet):
Displays the total number of accumulated workout and diet logs.

User Activity Trends (Past 7 Days):
Tracks user participation in workouts and diet logging over the past week.

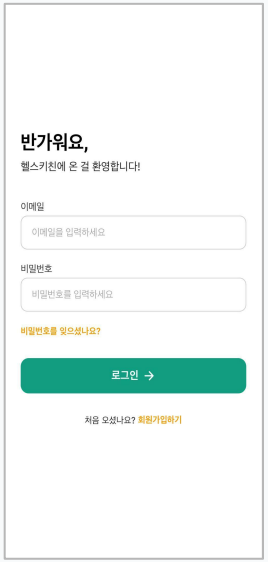
Top 10 Active Users:
Identifies the most active users, who may contribute meaningfully to AI training.

AI Training Data Distribution & Model Update Timestamp:
Confirms the current distribution of training data and when the model was last updated

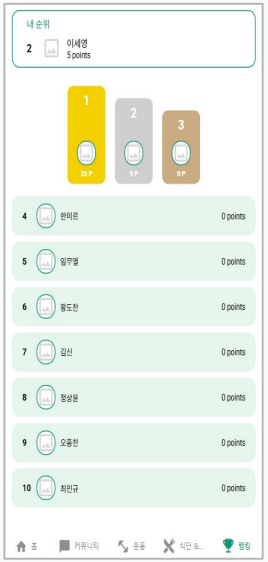
04 Final Outcomes

2) Project Demonstration – Mobile App (User Screenshots)

App Launch Screen

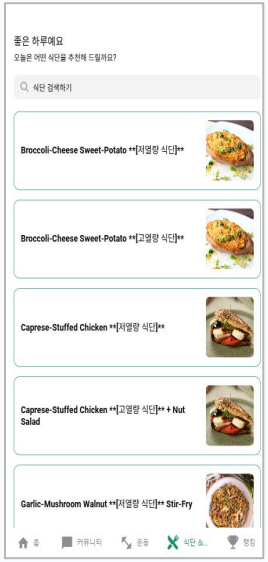


Login Screen

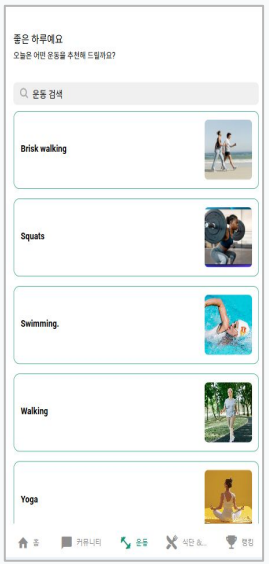


Ranking Screen

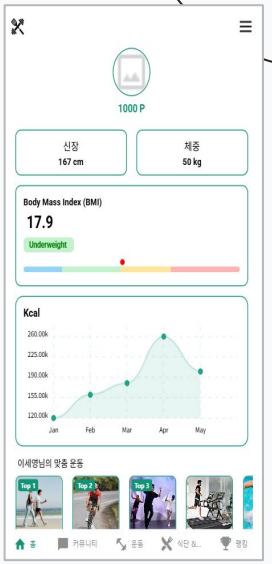
Community Screen



Diet & Recipe Tab



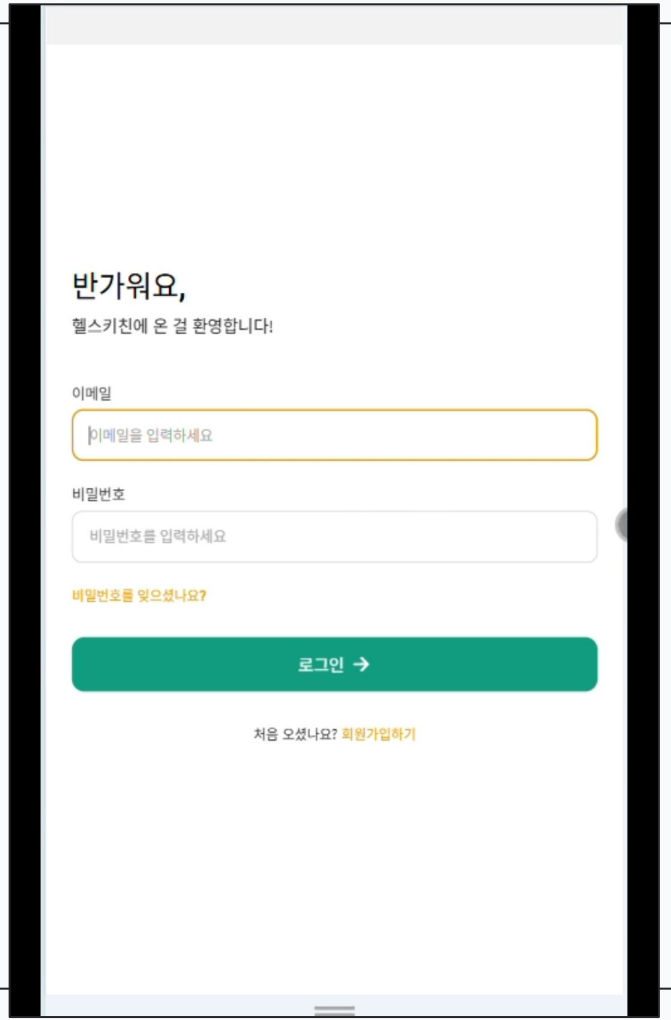
Workout Search Tab



Home Screen

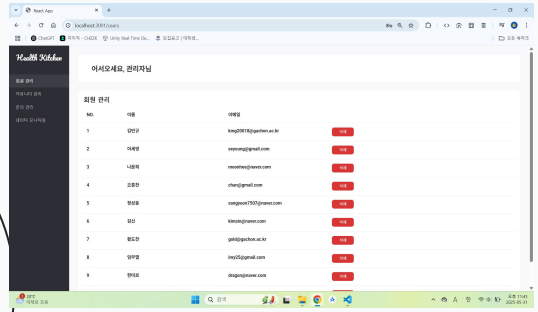
04 Final Outcomes

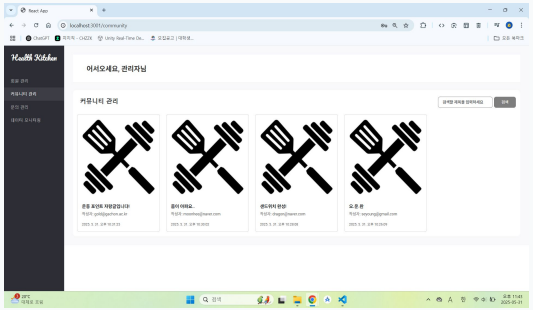
2) Project Demonstration – Mobile App (User)

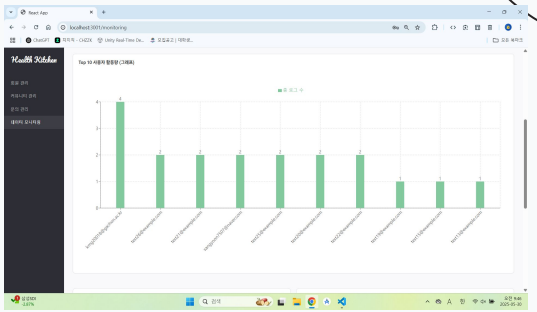


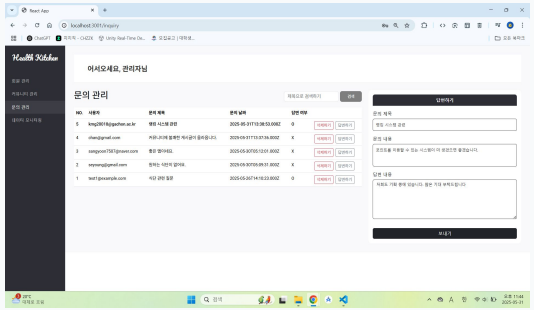
04 Final Outcomes

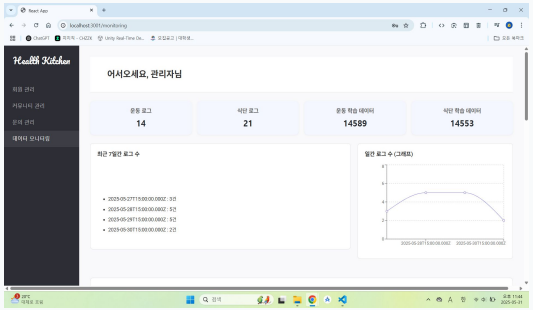
2) Project Demonstration - Web (Admin) Screenshots

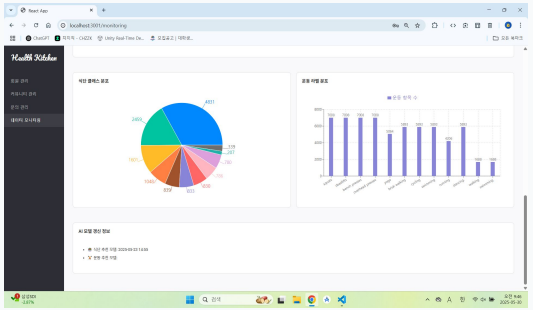






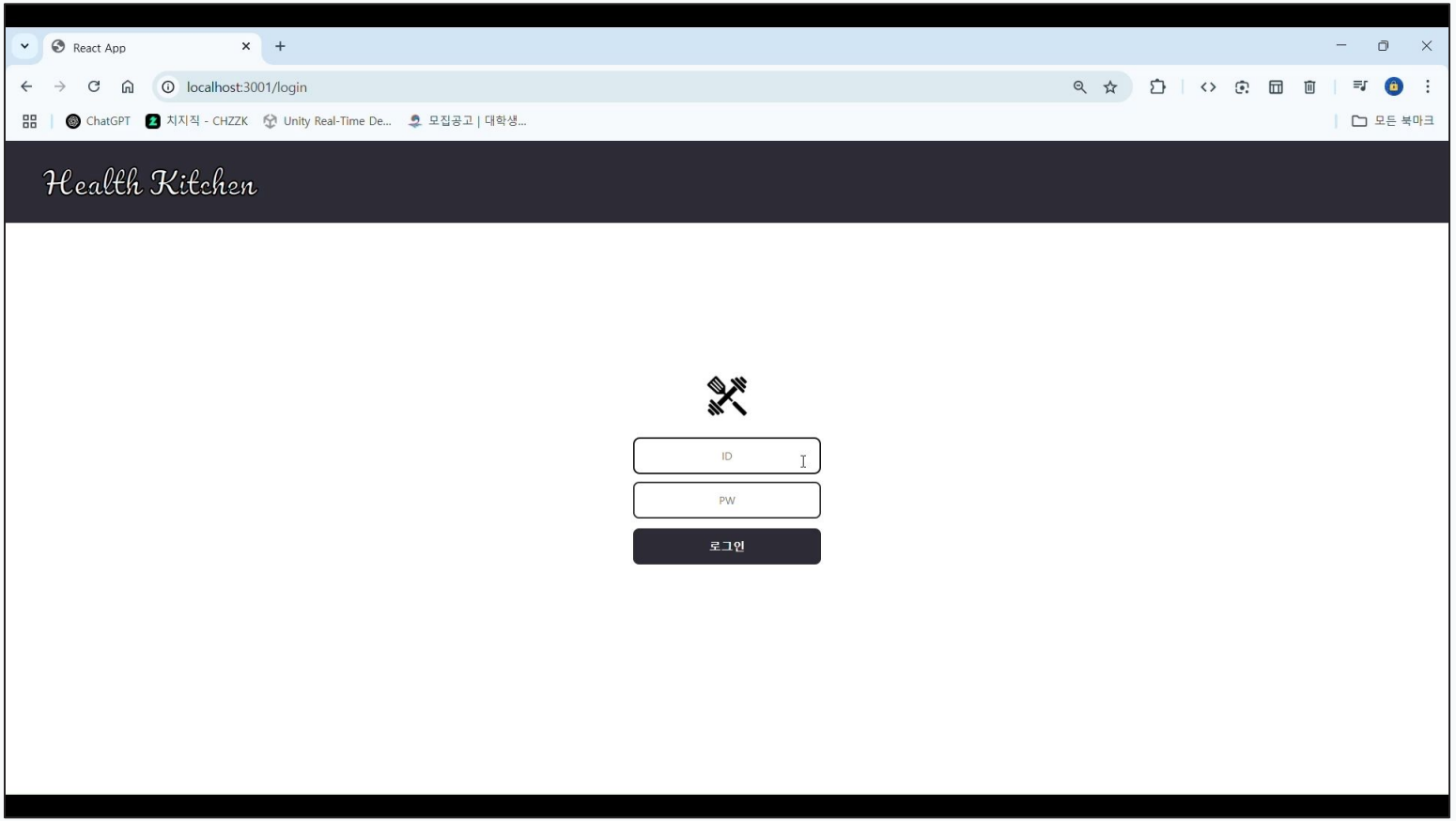






04 Final Outcomes

2) Project Demonstration - Web (Admin) Video



Fin.