BSCS FINAL PROJECT

NutriCarePro

Advisor: **Maria Nazir**

Presented by:
Group ID: F24BS170

Student Reg#	Student Name
L1F21BSCS0293	**M.Mubashir Qadeer**
L1F21BSCS0298	**Moeez-Ur-Rehman**
L1F21BSCS0573	**Mian Hamza Waheed**

Faculty of Information Technology

University of Central Punjab

NutriCarePro

By

M.Mubashir Qadeer
Moeez-Ur-Rehman
Mian Hamza Waheed

Project submitted to

Faculty of Information Technology,

University of Central Punjab,

Lahore, Pakistan.

In partial fulfillment of the requirements for the degree of

BACHELOR OF SCIENCE
IN
COMPUTER SCIENCE

Project Advisor	Manager Projects

Abstract

NutriCare Pro is a localized diet recommendation system designed to support Pakistani users, especially those suffering from chronic diseases such as diabetes, hypertension, and cardiovascular issues. Unlike international platforms, NutriCare Pro offers culturally relevant and economically accessible solutions by incorporating local food items, regional supplements, and health data. Using Django REST API, MongoDB/PostgreSQL, ReactJS, and ML models (TensorFlow), the system personalizes diet plans and health tracking while offering e-commerce integration for supplement purchases. This document outlines the system's technical design, component structure, interaction diagrams, testing strategies, and implementation progress.

Dedication

We dedicate this project to **our beloved parents**, whose endless support, prayers, and encouragement have been our greatest strength throughout this journey.

We also extend our gratitude to our **advisor, Ms. Maria Nazir**, for her constant guidance and valuable feedback.

Lastly, we dedicate this work to all those who strive to make a positive difference in the field of health and technology, especially those helping individuals manage chronic illnesses through innovation and care.

Acknowledgements

First and foremost, we are profoundly thankful to **Allah Almighty** for granting us the strength, patience, and determination to complete this project successfully.

We express our sincere gratitude to our respected project advisor, **Ms. Maria Nazir**, for her consistent guidance, encouragement, and insightful feedback throughout all phases of the Software Development Project. Her mentorship played a vital role in shaping the direction and quality of our work.

We would also like to thank the **Faculty of Information Technology and Computer Science at the University of Central Punjab** for providing us with the academic foundation and technical resources necessary to carry out this project.

Our heartfelt thanks go to our families for their unwavering support, patience, and motivation during challenging times.

Lastly, we appreciate the collaborative efforts and dedication of all **team members**, who contributed equally to every task, meeting each deadline with determination and professionalism. Without this teamwork, the successful completion of **NutriCare Pro** would not have been possible.

List of Figures

List of Tables

Table of Contents

Revision History

Name	Date	Reason For Changes	Version

Chapter 1. Introduction

1.1 Product

In Pakistan, poor dietary practices and limited nutrition education have led to rising rates of malnutrition, obesity, and non-communicable diseases like diabetes and heart conditions. Existing international diet apps are not tailored to the specific dietary habits, cultural preferences, or locally available foods in Pakistan, creating a disconnect between generic nutrition advice and practical implementation. Additionally, these apps often overlook user-specific health data, such as body metrics and fitness goals, and exclude affordable, locally sourced food options. NutriCare Pro addresses this gap by offering a web-based platform that combines health data utilization, localization, cultural relevance, and accessibility to deliver personalized diet plans based on Pakistani food items, promoting healthier and more practical dietary habits. Background.

1.2 Background

In an age where lifestyle diseases are on the rise, maintaining a healthy diet is crucial for improving overall health and well-being. However, many individuals in Pakistan face challenges in adopting balanced diets due to a lack of access to personalized nutrition advice tailored to their specific needs and cultural preferences. Existing diet-planning platforms often fail to address the unique requirements of local populations, relying instead on global food databases and generic recommendations.

NutriCare Pro is a web-based application developed to provide a solution to these challenges. By integrating users' health data—such as age, weight, height, and fitness goals—the platform generates personalized diet plans that are centered around locally available food products. This approach ensures cultural relevance, affordability, and practicality for the Pakistani population. NutriCare Pro not only simplifies the process of creating balanced diet plans but also empowers users to monitor their nutritional intake and track their progress toward a healthier lifestyle.

The project is built upon core areas of knowledge, including web application development, nutrition science, and data analytics, to offer a user-friendly and impactful solution to the growing nutritional challenges in Pakistan.

Pakistan faces a dual burden of malnutrition: while undernutrition and micronutrient deficiencies are common in rural and low-income areas, urban populations are increasingly affected by obesity and related lifestyle diseases such as diabetes and cardiovascular disorders. These issues stem from unhealthy dietary patterns, a lack of nutrition awareness, and limited access to affordable, practical diet-planning tools.

International nutrition apps, such as MyFitnessPal, have gained popularity but remain impractical for Pakistani users due to their reliance on global food databases and expensive, imported ingredients. Furthermore, these apps do not consider cultural dietary habits or socioeconomic limitations, making their recommendations less effective and often inaccessible.

1.3 Objective(s)/Aim(s)/Target(s)

- Create a localized food product database.
- Design a recommendation engine for personalized diet plans.
- Offer user-friendly interfaces for inputting dietary preferences and restrictions.

1.4 Scope

The software will:

- Provide customized diet plans.
- Use a database of local Pakistani products.
- Target both health-conscious users and those with dietary restrictions.
- Be accessible via a web or mobile application.

1.5 Business Goals

- Provide the Pakistani market with an inexpensive diet planning tool.
- Use familiar and easily accessible items to promote good eating habits.
- Establish a platform that is scalable for upcoming improvements.

1.6 Document Conventions

None.

1.7 Miscellaneous

< Provide any other information that you feel will help organize and conduct the project work.>

Chapter 2 Overall Description

2.1 Product Features

Personalized Health and Nutrition Advice

- Tailors dietary recommendations based on individual health data, focusing on managing chronic conditions such as diabetes, hypertension, and cardiovascular diseases.
- Utilizes machine learning to adapt advice over time as the user's health profile changes.

Progress Monitoring

- Tracks users' health and dietary habits, allowing them to monitor their improvement over time.
- Provides visual progress reports to encourage adherence to health goals.

Chronic Disease Management

- Specialized tools to assist users in managing specific chronic conditions.
- Offers condition-specific diet plans and guidance to improve health outcomes.

User-Friendly Web Platform

- Easy-to-navigate interface designed for a broad range of users, including those unfamiliar with technology.
- Secure and intuitive access to personal health data and recommendations.

Data-Driven Customization

- Collects and analyzes user data to continuously improve recommendation accuracy.
- Ensures privacy and security of sensitive health data.

Comprehensive Support System

- Offers educational resources on maintaining a healthy lifestyle.
- Provides notifications and reminders to help users stay on track with their health goals.

2.2 Functional Description

NutriCare Pro is a comprehensive health and nutrition management system designed to provide personalized dietary recommendations, chronic disease monitoring, and an integrated e-commerce platform for health-related products.

2.2.1.1 Core Functionalities:

1. **Personalized Nutrition Plans:**
    - Users receive customized diet plans based on their health data, preferences, and dietary goals.
    - AI-driven recommendations ensure personalized meal suggestions.

2. **Chronic Disease Management:**
    - Allows users to track and manage chronic conditions like diabetes, hypertension, and cardiovascular diseases.
    - Provides insights and alerts based on health trends.
3. **Health Progress Monitoring:**
    - Users can log and track their dietary intake, weight, BMI, and other health metrics.
    - Progress reports and analytics help users stay on track.
4. **Integrated Online Store:**
    - Offers health supplements, organic food, and wellness products.
    - Users can purchase recommended products based on their dietary needs.
5. **User Engagement & Assistance:**
    - Chatbots and virtual assistants for real-time dietary guidance.
    - Community forums and expert consultations for user support.

2.3 User Classes and Characteristics

Primary Users (Chronic Disease Patients)

- **Characteristics**:
    - Individuals managing chronic conditions such as hypertension and diseases.
    - Limited technical expertise; prefer an intuitive and user-friendly interface.
    - Frequent users rely on the platform for ongoing dietary recommendations and health tracking.
- **Key Functions**:
    - Access to personalized diet plans and progress tracking.
    - Notifications and reminders for health goals and recommendations.

Health Enthusiasts

- **Characteristics**:
    - Users without chronic conditions but interested in maintaining a healthy lifestyle.
    - Moderate technical expertise and familiarity with web-based applications.
    - Periodic users, leveraging features like nutrition advice.
- **Key Functions**:
    - General health recommendations.

Healthcare Providers and Nutritionists

- **Characteristics**:
    - Professionals seeking insights into their patients' dietary habits and progress.
    - High technical expertise; requires secure access to patient data and advanced analytics.
    - Occasional users focusing on specific patient cases.
- **Key Functions**:
    - Review patient progress and adapt recommendations.
    - Use patient data to create or refine personalized diet plans.

Caregivers and Family Members

- **Characteristics**:
    - Non-professional caregivers assisting patients with chronic conditions.
    - Minimal technical expertise; requires simplified views of patient data and recommendations.
    - Infrequent users, accessing the platform primarily for monitoring and assistance.
- **Key Functions**:
    - View and help implement dietary recommendations.
    - Monitor patient progress and adherence to health plans.

Administrators

- **Characteristics**:
    - Responsible for managing user accounts, security, and platform maintenance.
    - High technical expertise; requires access to backend functionality and data management tools.
    - Infrequent users, primarily handling issues or updates.
- **Key Functions**:
    - User account management and system maintenance.

2.4 Design and Implementation Constraints

Regulatory and Compliance Constraints

- The platform must comply with:
    - **HIPAA (Health Insurance Portability and Accountability Act)** for handling sensitive health data.
    - **GDPR (General Data Protection Regulation)** for managing user privacy, especially for users in the EU.

- Data storage and processing must meet local regulatory requirements for healthcare-related applications.

Database Constraints

- MongoDB is the designated database:
    - It requires schema-less design, which may impact the modeling of structured health data.
    - Sharding and replication mechanisms must be implemented to ensure scalability and fault tolerance.
- Data consistency and relationships must be managed programmatically due to MongoDB's NoSQL nature.

Security Considerations

- The platform must implement:
    - Secure authentication protocols (e.g., OAuth 2.0, JWT).
    - Role-based access control (RBAC) for user and admin privileges.
    - Data encryption at rest (e.g., MongoDB encrypted storage engine) and in transit (e.g., HTTPS with TLS).
- Regular security audits and adherence to best practices for preventing data breaches.

Performance Constraints

- The system must handle high-concurrency scenarios with thousands of users accessing data simultaneously.
- Recommendations based on machine learning models must respond within a latency threshold of 2-3 seconds.

Integration Constraints

- APIs must follow RESTful standards for compatibility and scalability.

Hardware Limitations

- The platform should operate efficiently on devices with minimum specifications (2 GB RAM, 2 GHz processor) to ensure accessibility for users with older hardware.
- Server-side operations must optimize memory usage and compute-intensive tasks, particularly for machine learning model processing.

Technology and Tooling Requirements

- The development stack is constrained to:
    - **Python Django** for backend development.
    - **MongoDB** for database management.
    - **React** for frontend development.
- Specific libraries and tools include TensorFlow or PyTorch for machine learning and Celery for asynchronous tasks.

Parallel Operations

- The system must support parallel execution of:
    - Machine learning model inference for multiple users.
    - Background tasks like health data updates and email notifications.

Design Standards and Conventions

- Must follow clean code practices and established programming standards to ensure maintainability.
- Use modular and reusable components to simplify future updates and scalability.

Maintenance and Deployment Constraints

- The system must be easy to maintain by the client organization, requiring well-documented code and infrastructure.

Communication Protocols

- The platform will rely on:
    - HTTPS for secure web communication.

2.5 Assumptions and Dependencies

The following assumptions and dependencies are identified for the successful implementation of the platform. These factors could affect the stated requirements if they change or are found to be incorrect:

2.5.1.1 Assumptions

1. **User Availability of Technology**
    - Users have access to devices with sufficient hardware capabilities (e.g., modern smartphones, tablets, or computers) and a stable internet connection.
    - Users are familiar with basic web navigation and usage of online platforms.
2. **External API Reliability**
    - Third-party APIs for fitness trackers, payment gateways, and health monitoring devices will remain operational and stable throughout the project lifecycle.
    - The APIs used provide accurate and timely data for integration.
3. **Data Quality and Availability**
    - Users provide accurate and complete health data for personalized recommendations.
4. **Scalability of MongoDB**
    - MongoDB will handle the expected volume of concurrent user interactions, data queries, and updates without significant performance degradation.
    - The database supports horizontal scaling and sharding to meet future growth.

5. **Regulatory Stability**
    - No significant changes occur in health data management regulations (e.g., GDPR, HIPAA) that would require significant rework of the system.
6. **Machine Learning Model Assumptions**
    - Pre-trained models used for recommendations are accurate and effective in providing personalized advice.
    - Training data is comprehensive and aligns with the targeted chronic diseases and nutritional guidance.

2.5.1.2 Dependencies

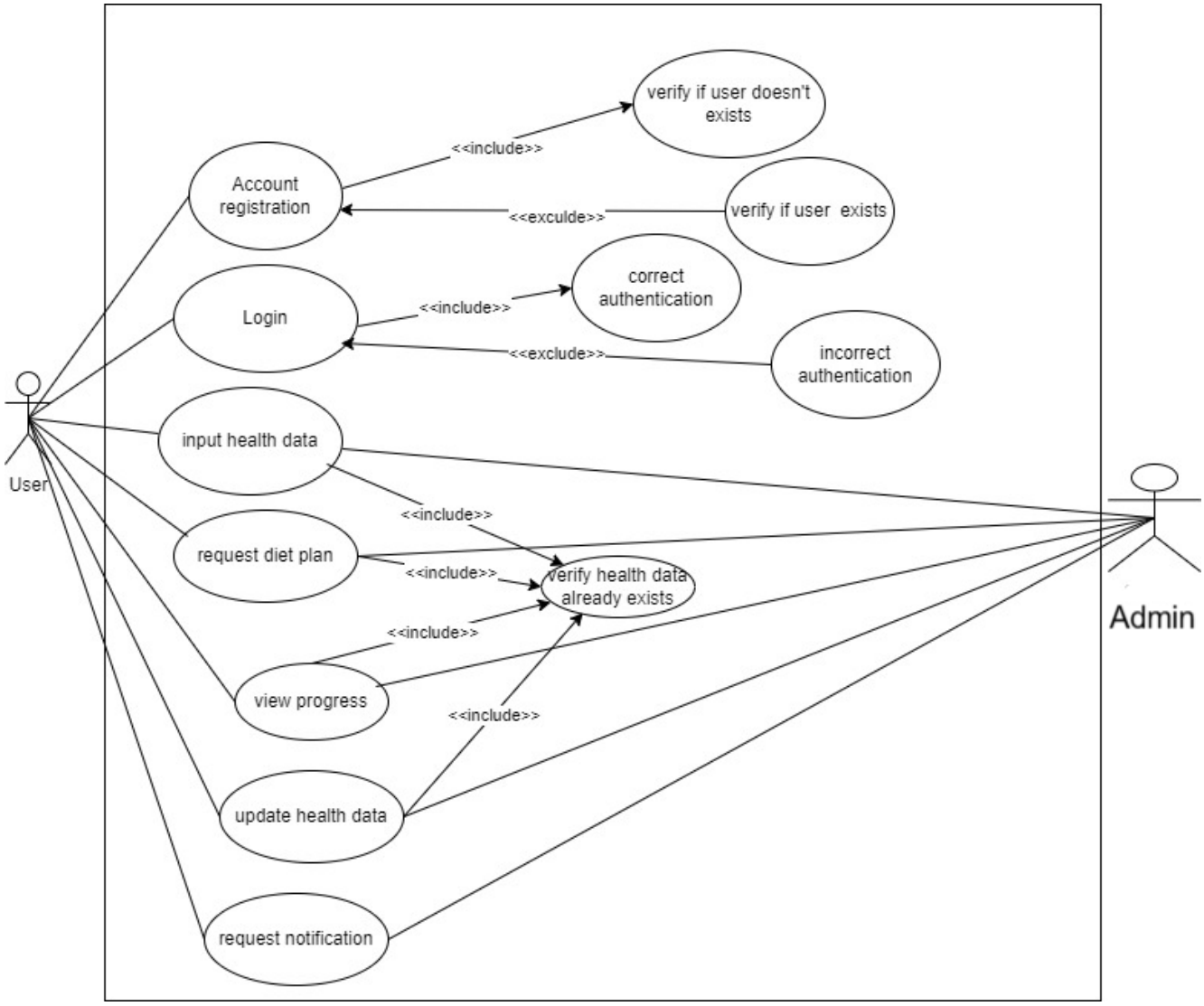
1. **Third-Party Services**
    - Reliance on third-party APIs for:
    - Payment processing (e.g., Stripe, PayPal).
    - Dependency on their uptime, updates, and API versioning.
    - Health and fitness data integration (e.g., Fitbit, Google Fit).
2. **Technology Stack**
    - The platform depends on the compatibility and performance of the selected stack, including Python, Django, MongoDB, and JavaScript frameworks (React or Vue.js).
    - Dependencies on libraries like TensorFlow or PyTorch for machine learning integration.
3. **Hosting Provider**
    - Cloud hosting services must support:
        - MongoDB setup (standalone or managed services like MongoDB Atlas).
        - Scalability for handling large datasets and high user traffic.
4. **Regulatory and Legal Compliance**
    - Dependency on legal teams or consultants to ensure compliance with global and local regulations concerning health data management.
5. **Team Expertise**
    - The development team is assumed to have expertise in the technologies and tools specified in the project.
    - Any training or knowledge gaps could delay project timelines.
6. **User Adoption**
    - The platform's success depends on its adoption by target user groups, including patients, healthcare providers, and caregivers.
    - Assumes user's willingness to input personal health data and trust the platform’s recommendations.

7. **Product Supplier Cooperation**

    - Health product suppliers must provide updated and accurate inventory details for the integrated online store.

Chapter 3 System Requirements

3.1 Functional Requirements



3.1.1 Use Case 1: User Registration

Identifier	UC-1	
Purpose	A user registers on the platform to create a personal account.	
Priority	High	
Pre-conditions	The user has access to the platform.	
Post-conditions	The user account is successfully created and stored in the database.	
Typical Course of Action		
S#	**Actor Action**	**System Response**
1	The user opens the registration page.	The system displays the registration form.
2	The user enters their details.	The system validates the inputs for format and completeness.
3	The user provides initial health data.	The system validates the health data.
4	The user submits the form.	The system creates the account and stores it in the database.
5		The system displays a confirmation message.
Alternate Course of Action		
S#	**Actor Action**	**System Response**
1	The user leaves mandatory fields empty.	The system prompts the user to fill in the missing fields.
2	The user enters an email that is already registered.	The system notifies the user about the existing email.
3		
…		

Table 1: UC-1

3.1.2 Use Case 2: User Login

Identifier	UC-2	
Purpose	A registered user logs in to access their account.	
Priority	High	
Pre-conditions	The user has already registered on the platform.	
Post-conditions	The user is logged in and redirected to the dashboard.	
Typical Course of Action		
S#	**Actor Action**	**System Response**
1	The user opens the login page.	The system displays the login form.
2	The user enters their email and password.	The system validates the credentials.
3	.	The system logs the user in and redirects to the dashboard.
Alternate Course of Action		
S#	**Actor Action**	**System Response**
1	The user enters incorrect credentials.	The system displays an error message.
2	The user clicks on "Forgot Password."	The system provides a password reset option.

3.1.3 Use Case 3: Health Data Submission

Identifier	UC-3
Purpose	The user submits health data for personalized recommendations.

Priority	Medium	
Pre-conditions	The user is logged in.	
Post-conditions	The data is securely stored in the database, and the system is updated with the user's information.	
Typical Course of Action		
S#	**Actor Action**	**System Response**
1	The user navigates to the health data page.	The system displays the health data submission form.
2	The user enters their health details..	The system validates the entered data.
3	The user submits the form.	The system securely stores the data in the database. The system displays a confirmation message.
Alternate Course of Action		
S#	**Actor Action**	**System Response**
1	The user enters invalid data.	The system prompts the user to correct the errors.

3.1.4 **Use Case 4: Personalized Diet Recommendation**

Identifier	UC-4
Purpose	The system provides tailored diet plans based on user health data.
Priority	Medium
Pre-conditions	• The user has logged in. • The user has submitted their health data.

Post-conditions	The user receives a diet plan recommendation on their dashboard.

Typical Course of Action		
S#	**Actor Action**	**System Response**
1	The user requests a diet recommendation.	The system retrieves the user's health data.
2		The system processes the data using a machine learning model.
3		The system displays a personalized diet plan on the dashboard.

Alternate Course of Action		
S#	**Actor Action**	**System Response**
1	The user has incomplete health data.	The system notifies the user to update their information.

3.1.5 Use Case 5: Progress Monitoring

Identifier	UC-5
Purpose	The user views progress over time to track health improvements.
Priority	Low
Pre-conditions	The user has logged in and submitted health data over multiple sessions.
Post-conditions	The user accesses a visual representation of their health progress..

Typical Course of Action		
S#	**Actor Action**	**System Response**
1	The user navigates to the progress section.	The system retrieves historical health data.
2		The system generates progress charts and insights.
3		

		The system displays the progress report to the user.
Alternate Course of Action		
S#	**Actor Action**	**System Response**
1	No historical data exists.	The system prompts the user to submit health updates.

3.1.6 Use Case 6: Health Data Update

Identifier	UC-6	
Purpose	The user updates their health data to ensure accurate recommendations.	
Priority	Medium	
Pre-conditions	The user is logged in.	
Post-conditions	The updated health data replaces the old data in the database.	
Typical Course of Action		
S#	**Actor Action**	**System Response**
1	The user navigates to the update page.	The system displays the health data update form.
2	The user modifies their health data.	The system validates the updated information.
3	The user submits the updates.	

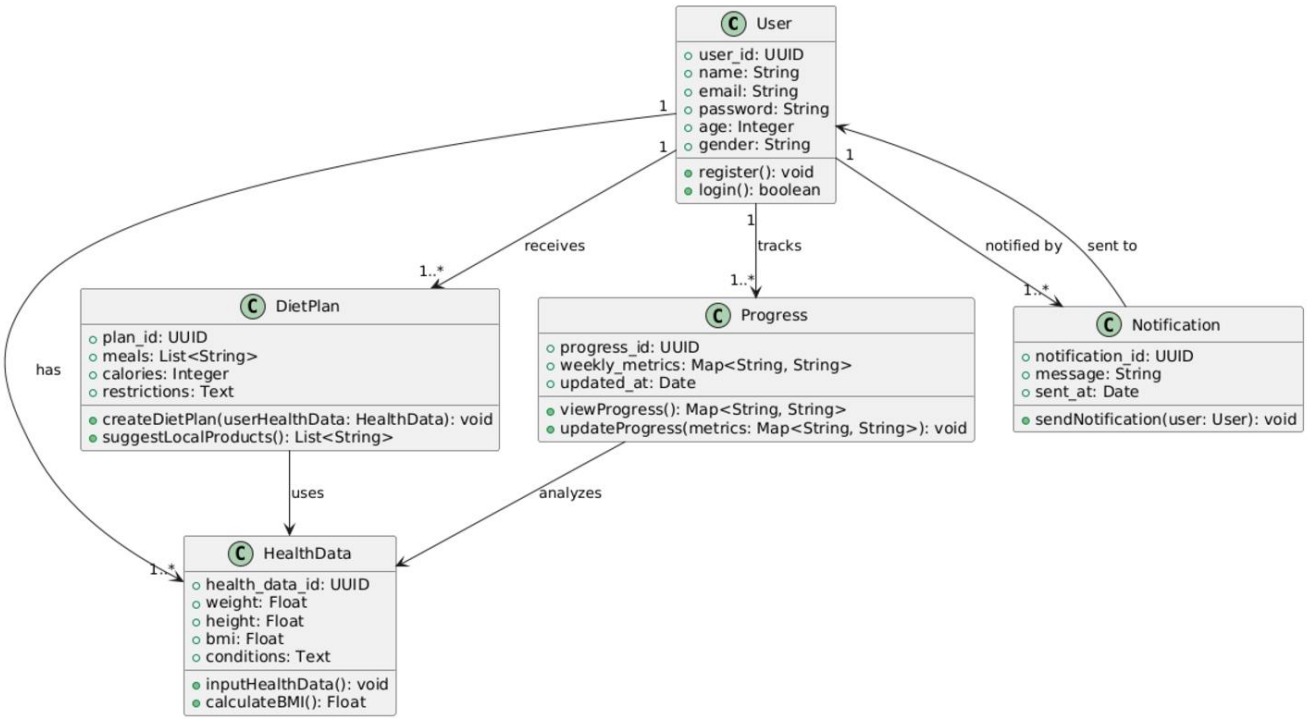
		The system stores the updated data in the database.
Alternate Course of Action		
S#	**Actor Action**	**System Response**
1	The user enters invalid data.	The system prompts the user to correct the errors.

3.1.7 Use Case 7: System Notification

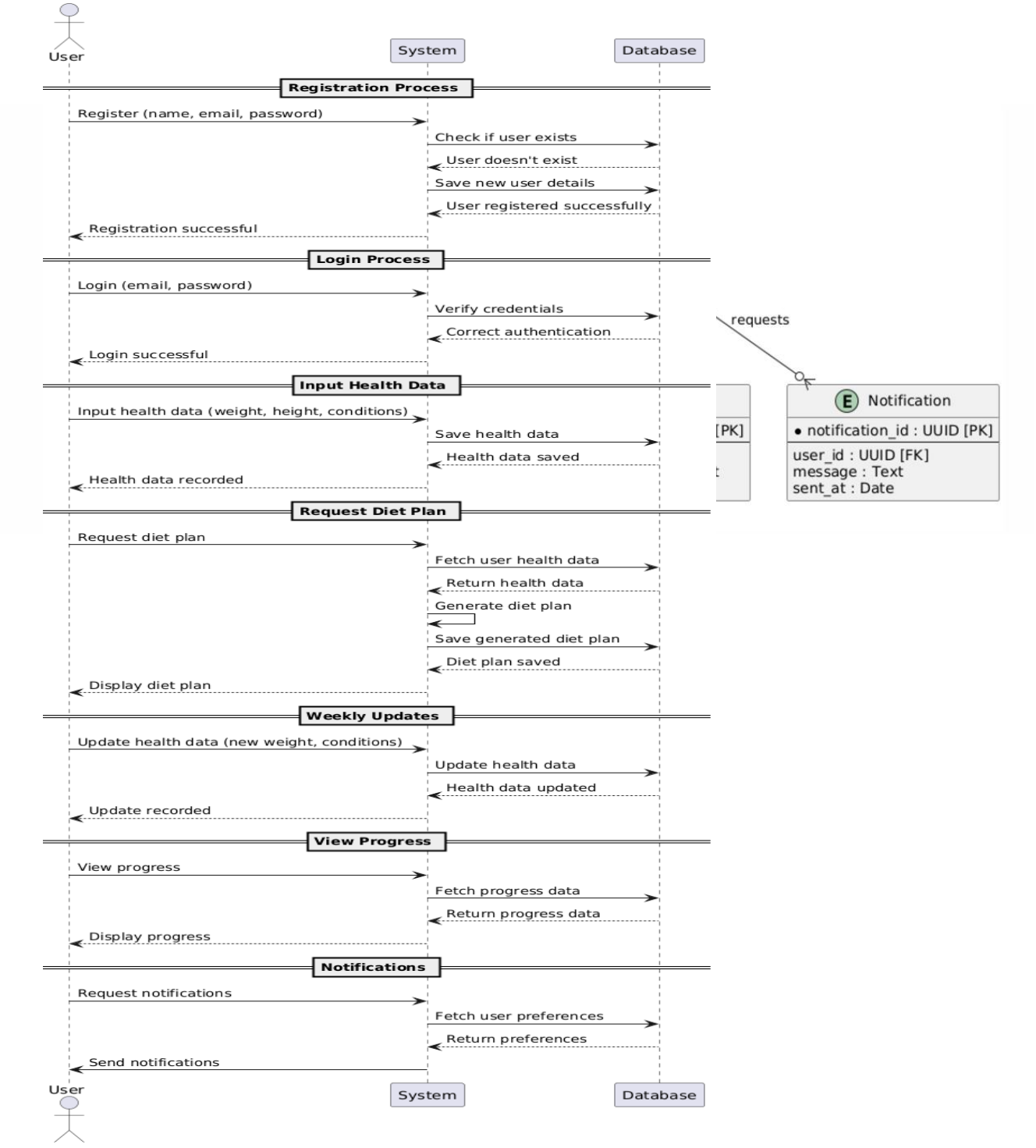
Identifier	UC-7	
Purpose	The system sends notifications to remind users to update their health data or check recommendations.	
Priority	Low	
Pre-conditions	The user has an active account.	
Post-conditions	The user receives a notification.	
Typical Course of Action		
S#	**Actor Action**	**System Response**
1		The system identifies users needing notifications.
2		The system generates and sends the notification.
Alternate Course of Action		
S#	**Actor Action**	**System Response**
1	The user has opted out of notifications.	The system skips sending notifications to the user.

3.2 Requirements Analysis and Modeling

Class Diagram:



Sequential Diagram:


User
System
Database
Registration Process
Register (name, email, password)
Check if user exists
User doesn't exist
Save new user details
User registered successfully
Registration successful
Login Process
Login (email, password)
Verify credentials
Correct authentication
Login successful
Input Health Data
Input health data (weight, height, conditions)
Save health data
Health data saved
Health data recorded
Request Diet Plan
Request diet plan
Fetch user health data
Return health data
Generate diet plan
Save generated diet plan
Diet plan saved
Display diet plan
Weekly Updates
Update health data (new weight, conditions)
Update health data
Health data updated
Update recorded
View Progress
View progress
Fetch progress data
Return progress data
Display progress
Notifications
Request notifications
Fetch user preferences
Return preferences
Send notifications
User
System
Database
requests
[PK]
Notification
• notification_id : UUID [PK]
user_id : UUID [FK]
message : Text
sent_at : Date

ERD Diagram:

3.3 Nonfunctional Requirements

3.3.1 Performance Requirements

1. Response Time

- **Diet Plan Suggestions**: The system must generate personalized diet plans within **5 seconds** of receiving user health data.
- **Progress Report Generation**: Progress reports should be displayed to users within **3 seconds** of the request.
- **Notifications**: Notifications must be delivered to users within **2 seconds** after the system triggers them.
- **Health Data Updates**: Updates to health data should reflect in the system and reports within **1 second** after submission.

2. Scalability

- The system must support up to **10,000 simultaneous users** with minimal degradation in performance.
- MongoDB database queries must handle a load of **50 concurrent requests per second** efficiently.

3. Data Storage and Retrieval

- Health records and historical data must be retrievable from the MongoDB database within **2 seconds**, regardless of the size of the dataset.
- The system must support storing at least **10 years of health data** for each user without impacting retrieval speed.

4. Availability

- The platform must maintain **99.9% uptime**, ensuring availability for users to access diet recommendations, update data, and monitor progress.

5. Resource Utilization

- The system must consume less than **70% CPU utilization** and **80% memory utilization** under peak load conditions on a standard server configuration (e.g., 16GB RAM, 8-core processor).

6. Rationale

- These requirements ensure the system remains responsive and reliable, delivering a positive user experience and enabling timely health recommendations critical for chronic disease management.

3.3.2 Safety Requirements

1. Data Integrity

- The system must prevent unauthorized access to user health data by implementing robust **encryption standards (e.g., AES-256)** for data storage and transfer.
- Health data must be validated before processing to prevent incorrect or misleading recommendations caused by invalid or incomplete inputs.

2. System Reliability

- The system must be thoroughly tested to avoid crashes or data loss during user interactions.
- Regular backups of health data must be maintained to allow restoration in case of system failure, with a backup frequency of **once every 24 hours**.

3. Privacy Safeguards

- Personal and health-related data must comply with relevant regulations such as **HIPAA (Health Insurance Portability and Accountability Act)** or equivalent local regulations.
- User consent must be obtained before collecting, processing, or sharing health data.

4. User Recommendations

- Diet plans and recommendations must be based on reliable, medically reviewed algorithms to avoid harm caused by incorrect or generic advice.
- The system must issue a disclaimer stating that the recommendations are not a substitute for professional medical advice.

5. Fail-Safe Mechanisms

- The system must gracefully handle unexpected errors by providing meaningful error messages without revealing sensitive information.
- In case of a failure in generating a diet plan, the system must notify the user and log the incident for review.

6. Safety Certifications

- The product must adhere to any relevant safety or medical software certifications applicable to its region of operation, ensuring it meets established health technology standards.

7. Restricted Features

- The system must not suggest diet plans to users with incomplete or invalid health data to avoid potentially harmful advice.

3.3.3 Security Requirements

1. User Authentication

- Users must authenticate using a **secure login process**, including:
    - **Password-protected accounts** with requirements for strong passwords (minimum eight characters, including uppercase, lowercase, numeric, and special characters).
    - Optional **two-factor authentication (2FA)** for added security.
- Passwords must be stored using secure **hashing algorithms (e.g., bcrypt)** to prevent unauthorized access.

2. Access Control

- Role-based access control (RBAC) must be implemented to ensure:
    - **Regular Users** can only access their health data and features.
    - **Administrators** have restricted access to manage the system, but cannot view sensitive user data without explicit permissions.
- Health data submitted by users must only be accessible to authorized components of the system for processing and recommendations.

3. Data Protection

- All user data must be encrypted using **AES-256** for storage and **TLS 1.3** for transmission.
- MongoDB must implement **role-based database access** to prevent unauthorized modifications or data leaks.
- Health data must be anonymized where applicable to enhance privacy.

4. System Hardening

- The application and server infrastructure must be hardened against attacks, including:
    - Implementation of **firewalls** and **intrusion detection/prevention systems (IDS/IPS)**.
    - Regular **security updates and patches** for all system components.
    - Protection against **SQL injection**, **cross-site scripting (XSS)**, and **cross-site request forgery (CSRF)** attacks.

5. Privacy Compliance

- The system must comply with privacy regulations such as:
    - **GDPR (General Data Protection Regulation)** for data privacy in Europe.

  - **HIPAA** for handling sensitive health information in applicable regions.
- A **privacy policy** must clearly outline how user data is collected, used, stored, and shared.

6. Incident Management

- In the event of a security breach:
  - Users must be notified within **72 hours** as per GDPR and similar regulations.
  - An audit log must capture all administrative actions and suspicious activities for review.

7. Security Certifications

- The platform should meet recognized security standards such as **ISO/IEC 27001** for information security management.

8. Session Management

- User sessions must expire after **15 minutes of inactivity**.
- Tokens for session authentication (e.g., JWT) must be secured and refreshed regularly to prevent replay attacks.

3.3.4 Additional Software Quality Attributes

1. Adaptability

- The platform should adapt to changes in user health data, automatically updating diet recommendations and progress reports.
- The machine learning model must be capable of incorporating new datasets and retraining with minimal developer intervention.

2. Availability

- The system must ensure **99.9% uptime** to guarantee uninterrupted access for users managing chronic conditions.
- Scheduled maintenance should be limited to off-peak hours and notified to users at least **24 hours in advance**.

3. Correctness

- Diet and progress recommendations must be accurate, medically sound, and validated by domain experts.
- Automated tests must ensure that all functionalities behave as intended under everyday and edge-case scenarios.

4. Flexibility

- The system should support integration with future enhancements, such as wearables or third-party health apps.
- Developers must be able to extend existing modules (e.g., adding new health parameters) without significant restructuring.

5. Interoperability

- The platform must integrate seamlessly with external APIs and services, such as fitness trackers and healthcare databases.
- It should support multiple data formats like JSON, CSV, and XML for importing/exporting health data.

6. Maintainability

- The codebase must adhere to industry best practices, including:
    - Modular and well-documented code.

  - Use of version control systems (e.g., Git).
- Developers should be able to fix bugs or make updates within **1-2 hours** for minor issues and **1-2 days** for major ones.

7. Portability

- The platform must be deployable across multiple environments, including cloud servers (AWS, Azure) and on-premises setups.
- The front end must be compatible with all modern browsers (Chrome, Firefox, Edge, Safari).

8. Reliability

- The system must handle up to **10,000 concurrent users** without failure.
- It should recover from unexpected failures (e.g., database disconnection) within **30 seconds**.

9. Reusability

- Core modules, such as the recommendation engine, must be designed for reuse in similar health-related applications.
- Shared components (e.g., user authentication) should follow standardized design patterns.

10. Robustness

- The system must gracefully handle invalid inputs and unexpected user behaviors, providing meaningful error messages.
- Database transactions must ensure data consistency even during partial failures.

11. Testability

- The platform must achieve **95% test coverage**, with automated tests for all critical functionalities.
- Unit and integration tests should be run before every deployment to ensure system stability.

12. Usability

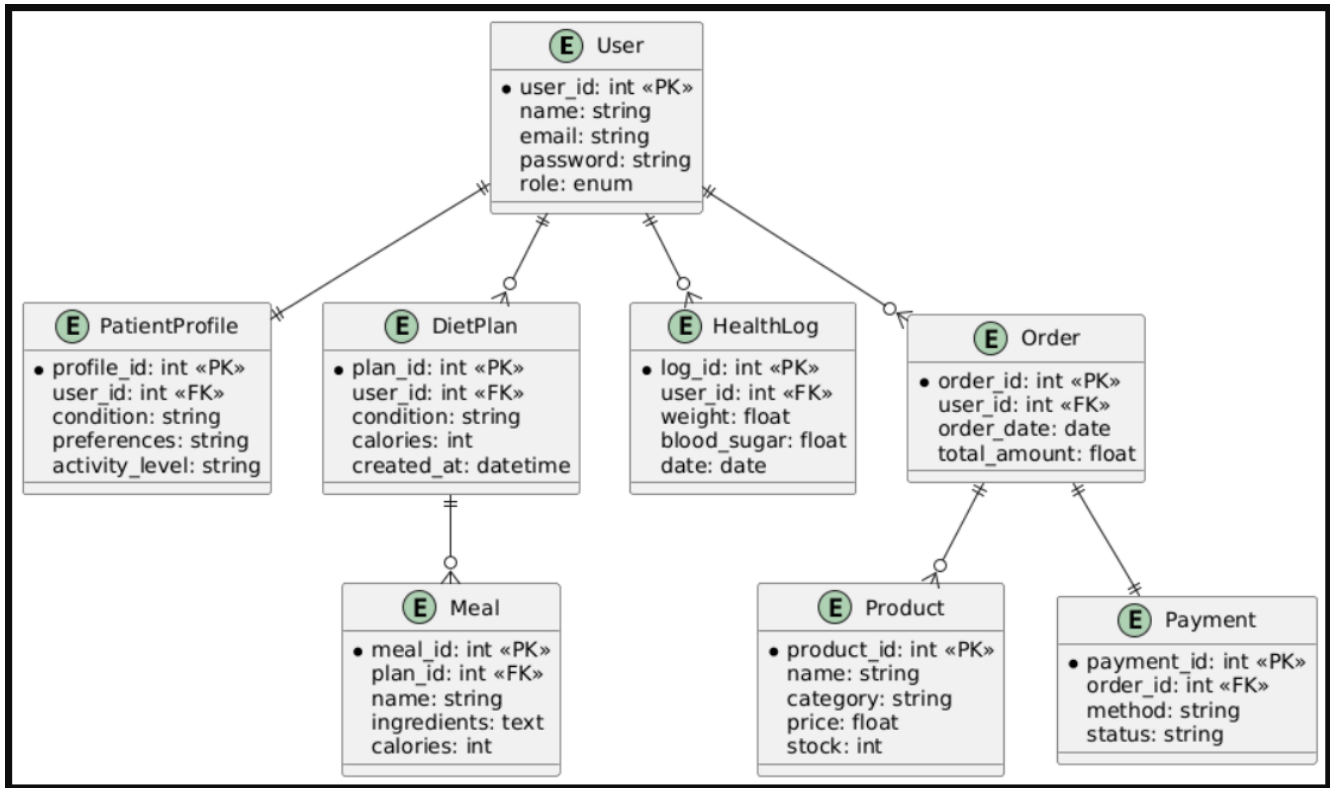
- The system must prioritize ease of use with:
  - A clean, intuitive user interface suitable for users with minimal technical expertise.
  - A learning curve of no more than **30 minutes** for first-time users.
- The platform should provide accessibility features (e.g., screen reader support, high contrast mode).

Chapter 4. Technical Architecture

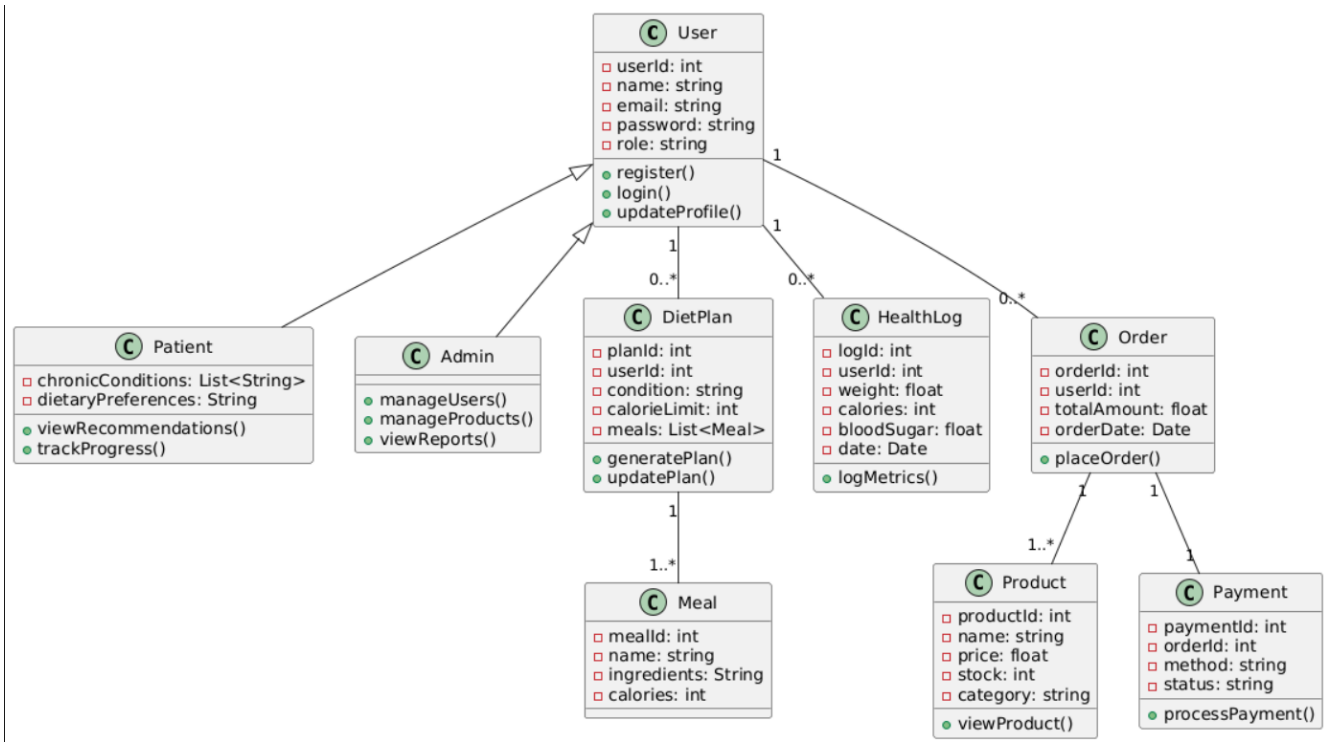
4.1 Application and Data Architecture

- **Architecture**: 3-Tier Architecture (Frontend – Backend – Database)
- **Frontend**: React.js
- **Backend**: Django REST Framework (Python)
- **Database**: MongoDB for flexible diet data + PostgreSQL for transactional data
- **Hosting**: AWS EC2, Dockerized containers, S3 for assets

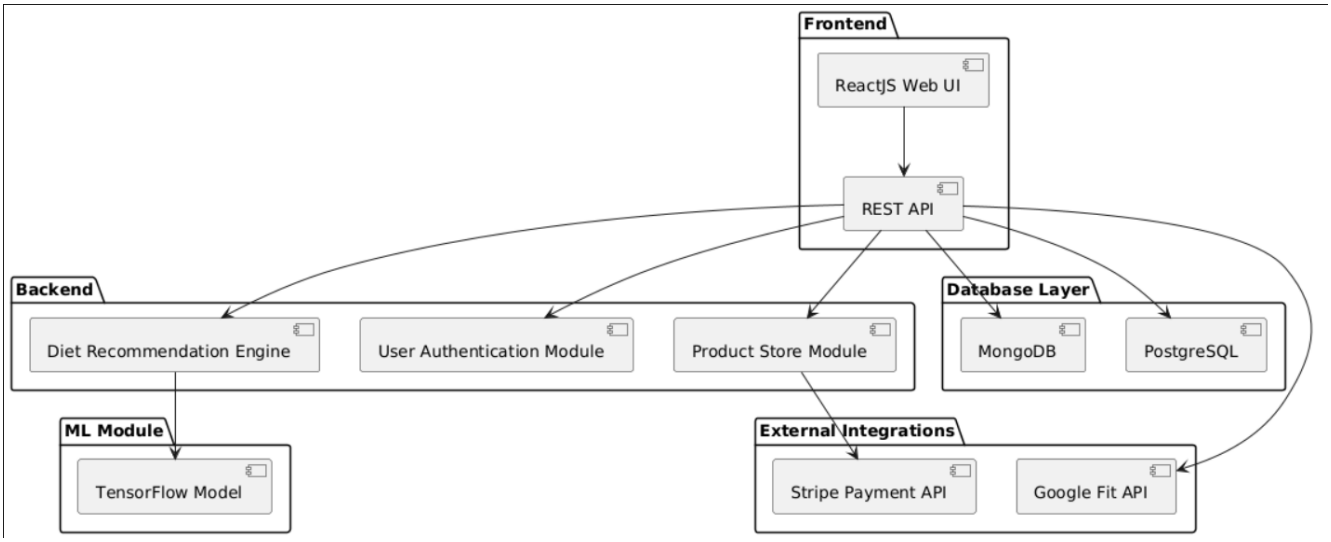
ERD Diagram:



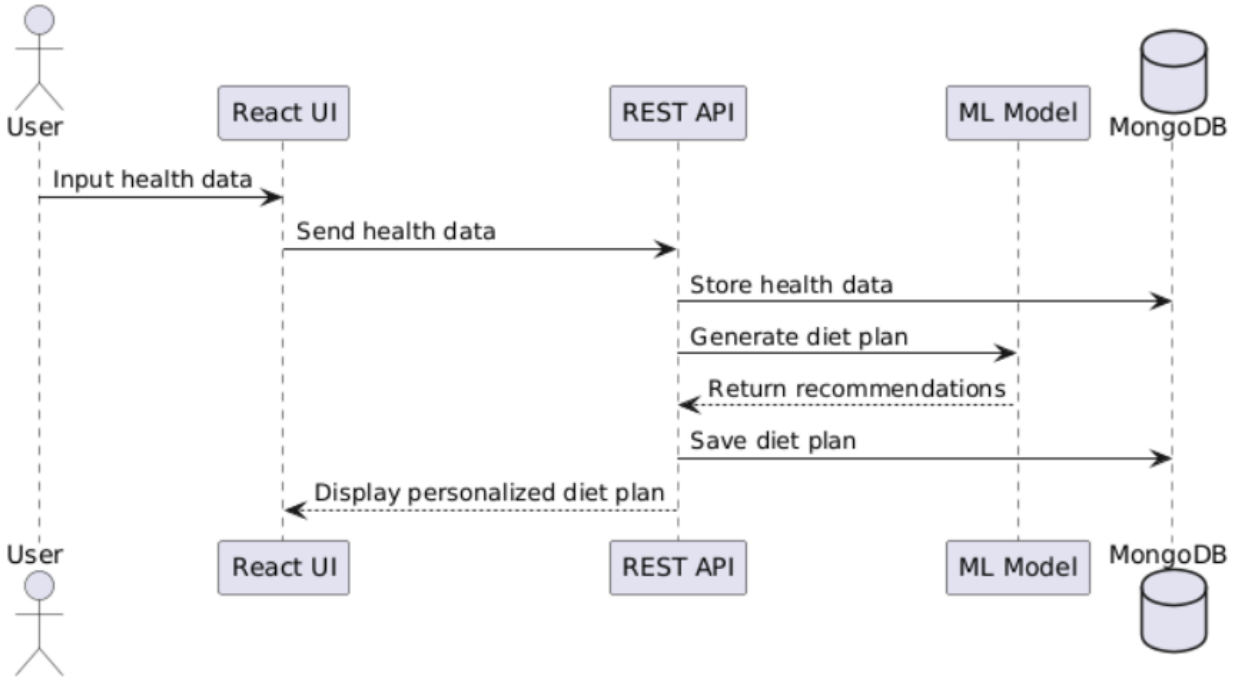
Class Diagram:



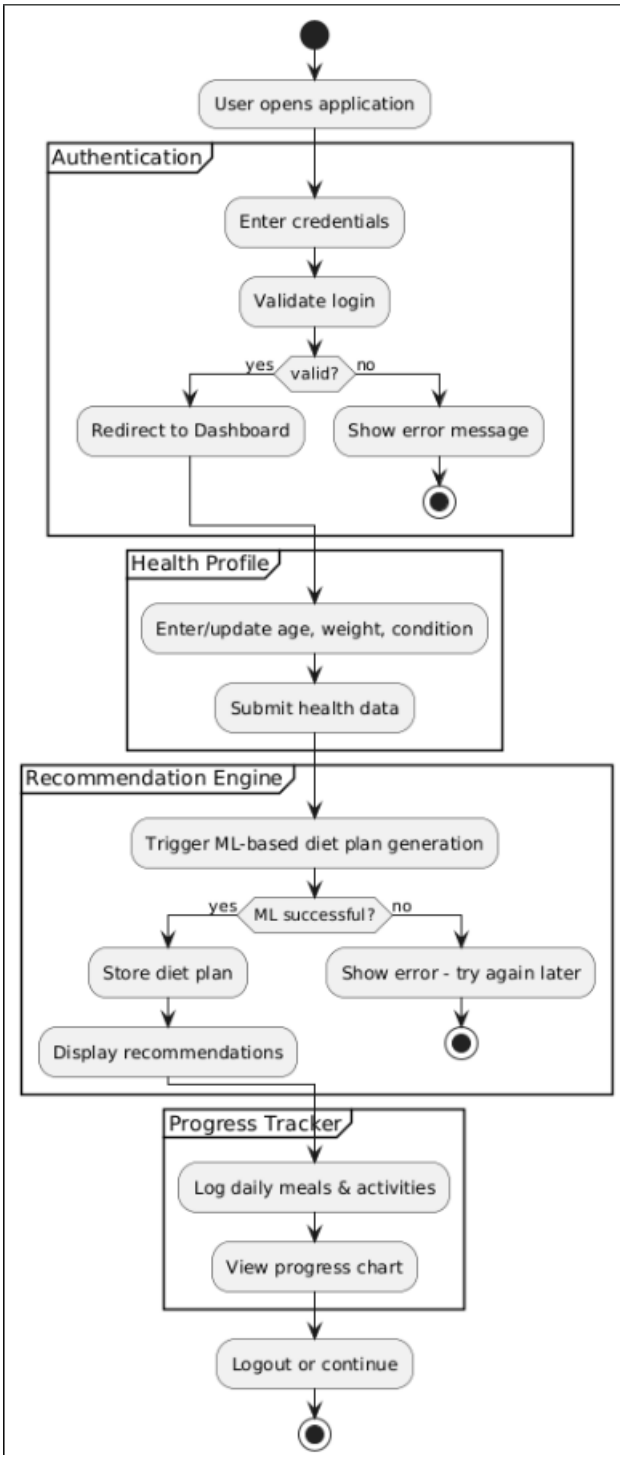
Component Diagram:



Sequence Diagram:



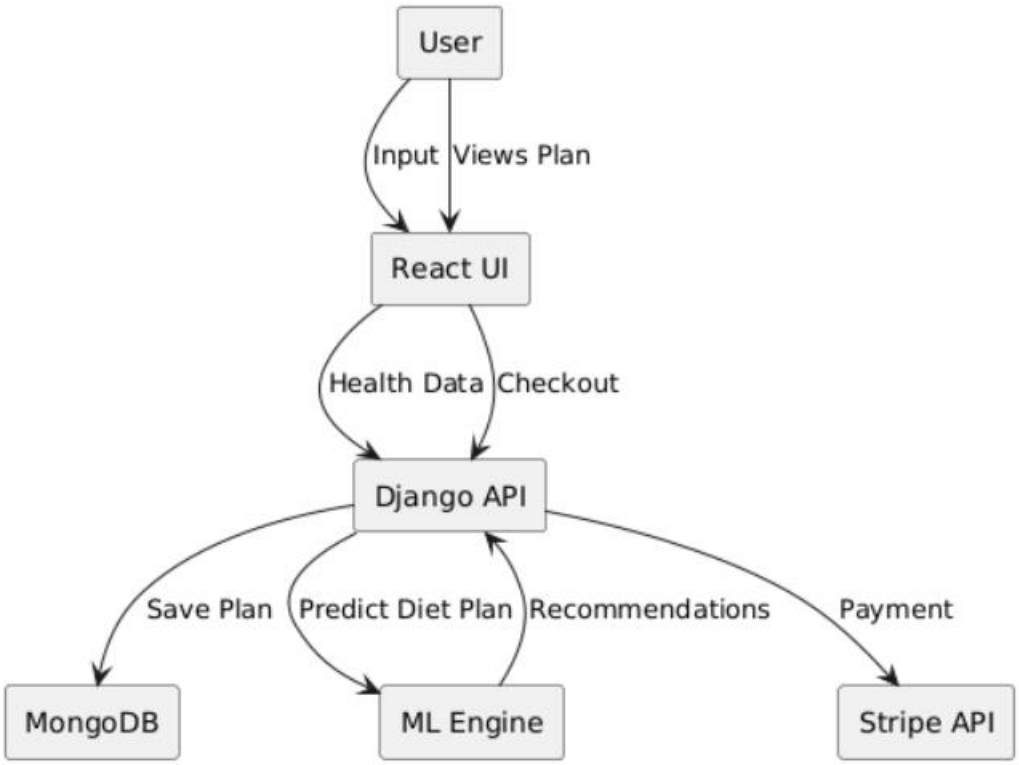
Activity Diagram:



4.2 Component Interactions and Collaborations

- User interacts via UI → data sent to backend
- Backend handles business logic, API responses
- ML engine generates diet suggestions
- Recommendations, analytics returned to the frontend

DFD Level 1:



4.3 Design Reuse and Design Patterns

Reuse:

- React components (Material UI)
- Django auth system
- Food data APIs
- Pre-trained ML models

Design Patterns:

- MVC (Backend)
- Strategy Pattern (Diet logic variations)
- Observer (Notifications)
- Singleton (Database connection pool)
- Factory (Diet plan creation logic)

4.4 Technology Architecture

Category	Stack/Tool
Frontend	React.js, Tailwind
Backend	Django REST Framework
Database	MongoDB, PostgreSQL
ML Framework	TensorFlow
Hosting	AWS EC2, MongoDB Atlas, Docker
Security	JWT, OAuth 2.0, HTTPS
Integrations	Google Fit, Stripe

4.5 Architecture Evaluation

Why Django? Fast API development, scalable, and secure
Why MongoDB? Flexible schema, ideal for dietary patterns
Why React? Component-based UI, excellent performance
Alternative Considered: Node.js (chosen Django for Python ML compatibility), Firebase (less suitable for complex backend logic)

Chapter 5. Detailed Design and Implementation

5.1 Component-Component Interface

All components communicate through RESTful endpoints, with JSON payloads as the standard data exchange format.

Key Interactions:

- User Interface → REST API: Sends health data, login credentials, and requests
- REST API → Diet Engine: For personalized meal recommendations
- REST API → DB: For persistent storage (via ORM + direct queries)
- REST API → Notification System: For trigger-based alerts
- Admin Panel → Product API: For CRUD operations on product listings

Sequence Diagram: Used to represent data flow across components (included earlier)

5.2 Component-External Entities Interface

NutriCare Pro interfaces with several third-party systems:

- **Stripe Payment Gateway**: Secure transaction processing
- **Google Fit API**: To fetch user activity data (steps, calories)
- **Food Nutrition API**: To pull detailed macronutrient info

All integrations follow **OAuth 2.0 or API Key** standards with **token refresh** mechanisms for stability.

Security: JWT is used for secure component communication.

5.3 Component-Human Interface

The application provides a **clean, responsive UI** focused on accessibility and clarity. Key interaction points include:

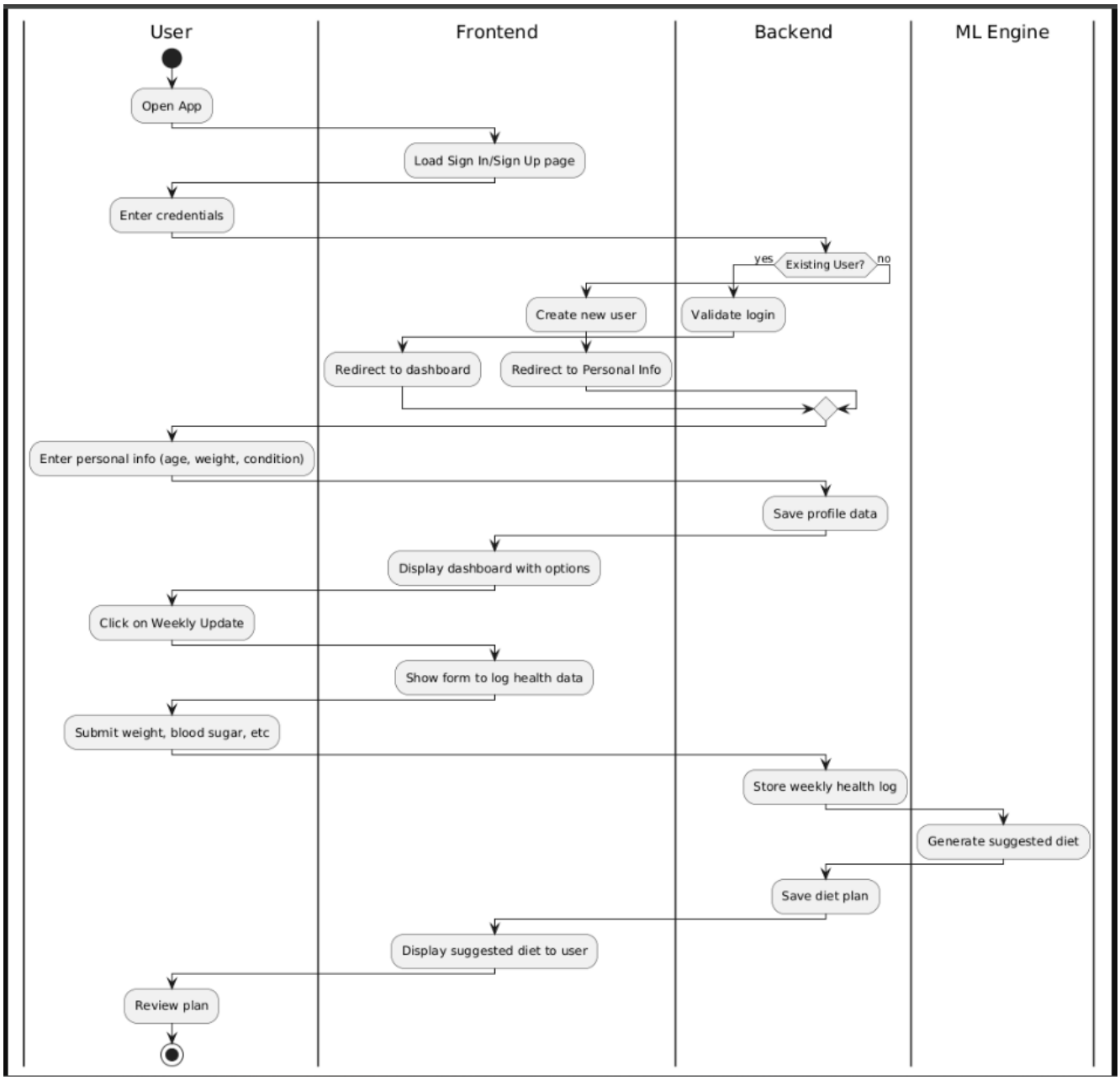
Screen	Description
Sign-up / Login	Email-password-based access
Health Data Form	Captures weight, conditions, and preferences
Dashboard	Shows diet plan, analytics, charts
Reccomendation	Recommend a diet plan
Profile	User Profile

HCI Norms Used:

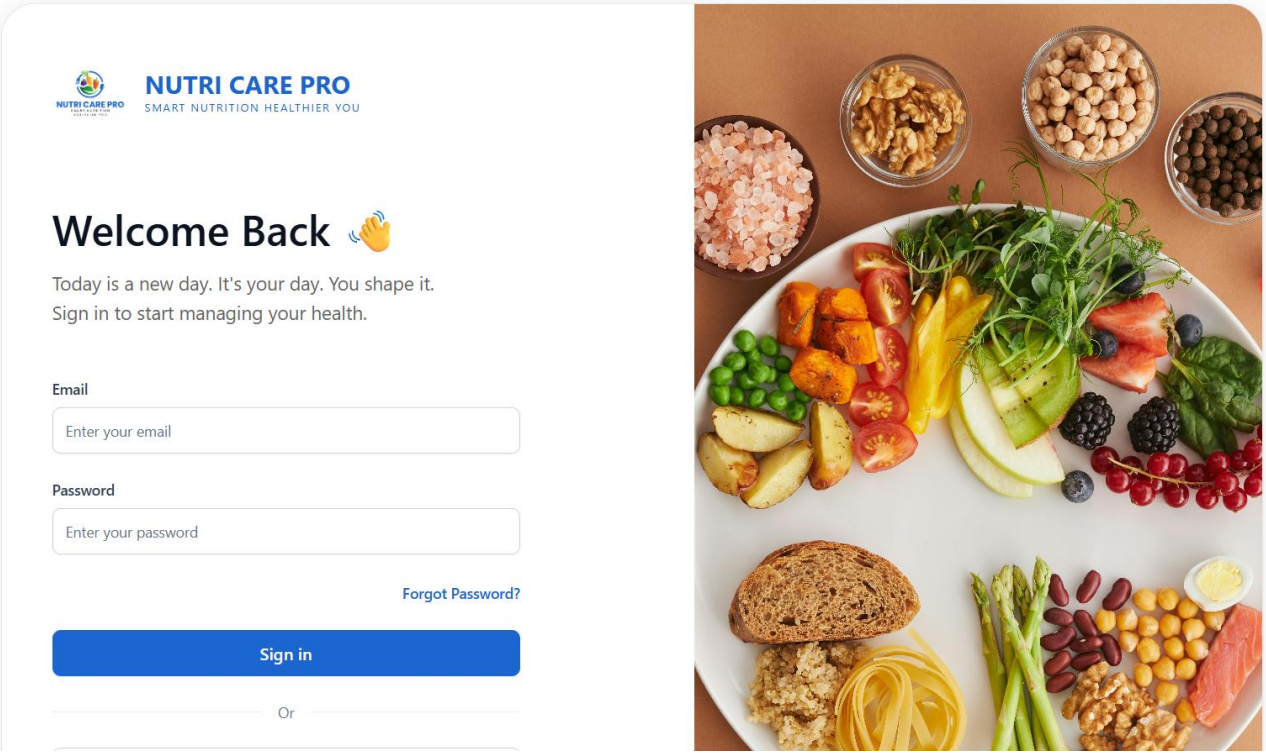
- Tab-friendly navigation
- Accessible color contrast
- Large buttons/icons for chronic patients
- Alert dialogs and progress feedback

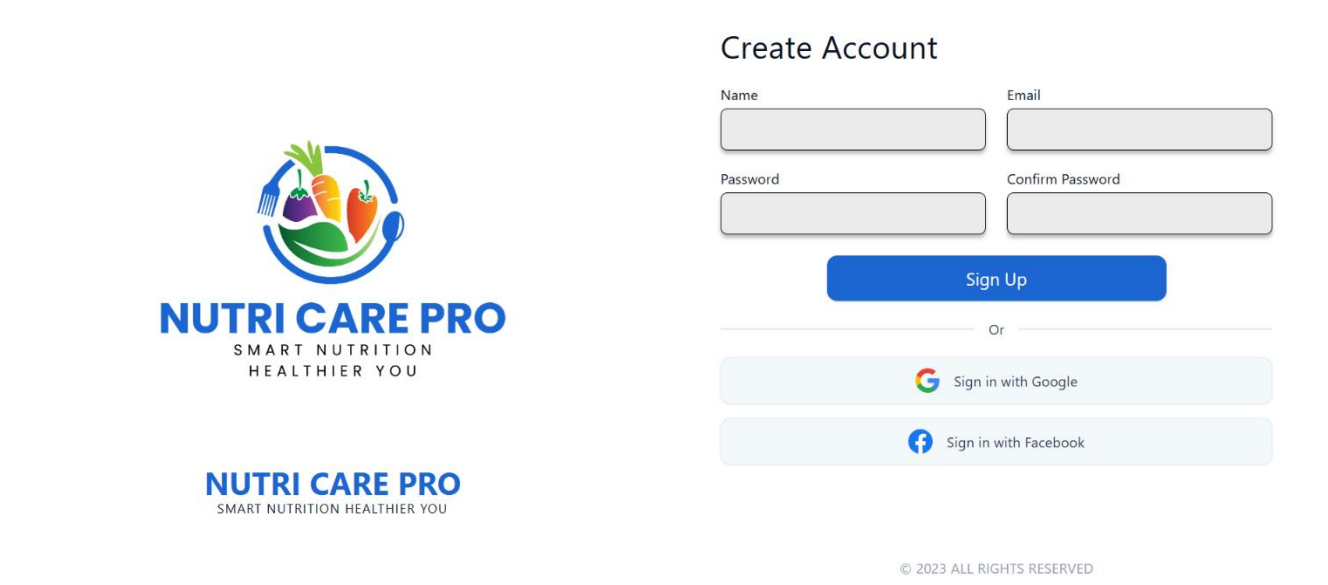
5.4 Workflow screenshot/prototype

The workflow of NutriCare Pro is initiated when a user signs up or signs in. Upon successful login, the user is redirected to the dashboard, where they can input personal health information or weekly updates. The system stores this data and invokes the machine learning model to generate a personalized diet plan, which is then displayed on the suggested diet page. This workflow ensures a seamless experience for chronic disease management, with real-time updates and interactive health monitoring.



5.5 Screens


NUTRI CARE PRO
SMART NUTRITION HEALTHIER YOU
Welcome Back
Today is a new day. It's your day. You shape it.
Sign in to start managing your health.
Email
Enter your email
Password
Enter your password
Forgot Password?
Sign in
Or


NUTRI CARE PRO
SMART NUTRITION
HEALTHIER YOU
NUTRI CARE PRO
SMART NUTRITION HEALTHIER YOU
Create Account
Name
Email
Password
Confirm Password
Sign Up
Or
Sign in with Google
Sign in with Facebook
© 2023 ALL RIGHTS RESERVED

NUTRI CARE PRO
SMART NUTRITION HEALTHIER YOU

Dashboard Weekly Update Profile Suggested Diet

Health Overview

Jan 22, 2025

Blood Sugar

80 mg/dL

Normal

Heart Rate

98 bpm

Normal

Blood Pressure

102 /72 mmHg

Normal

Activity Growth Jan 2025

BMI Calculator Last Week

Height

170 cm

Weight

72 kg

15 18.5 25 30 40

You're Healthy

Body Measurements

Last checked 2 Days Ago

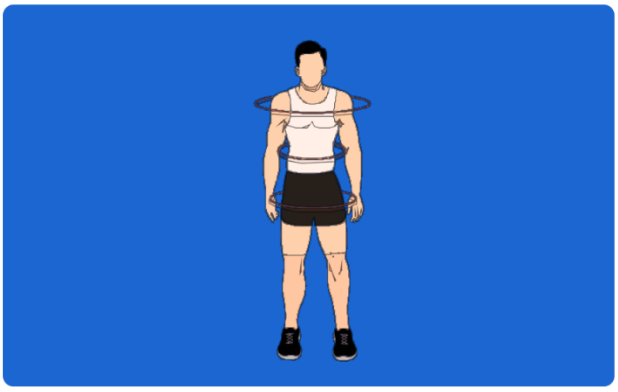
Chest (in)
44.5 ↑

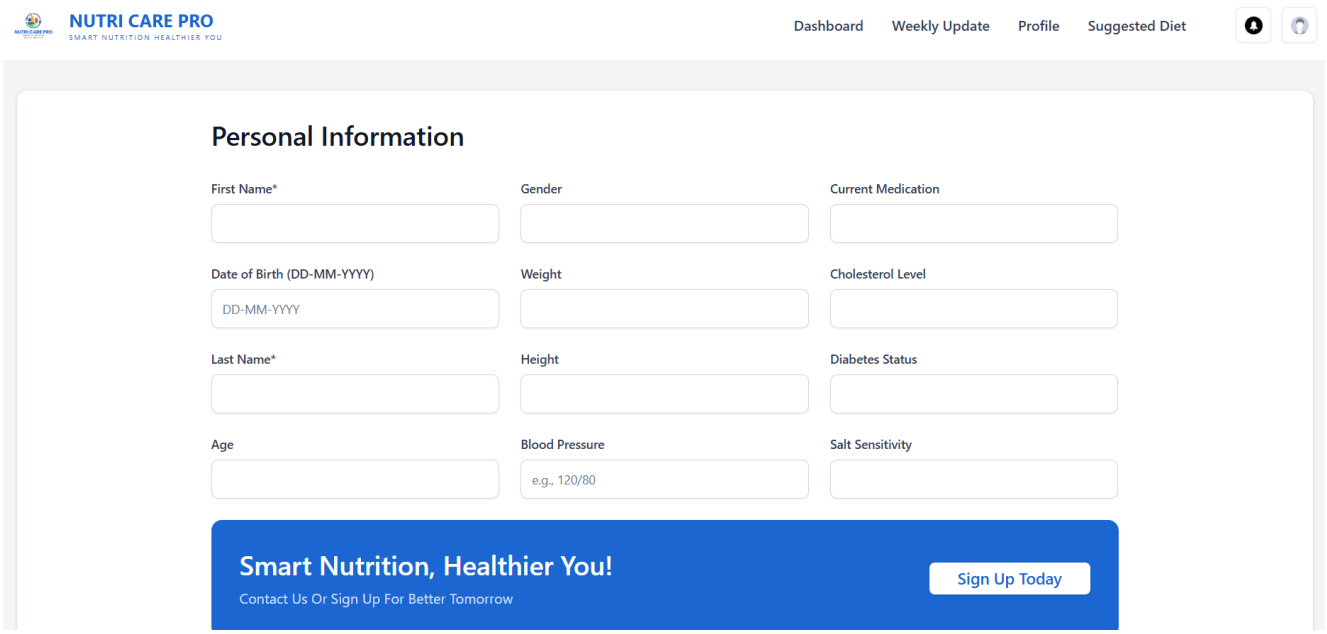
Waist (in)
34 ↓

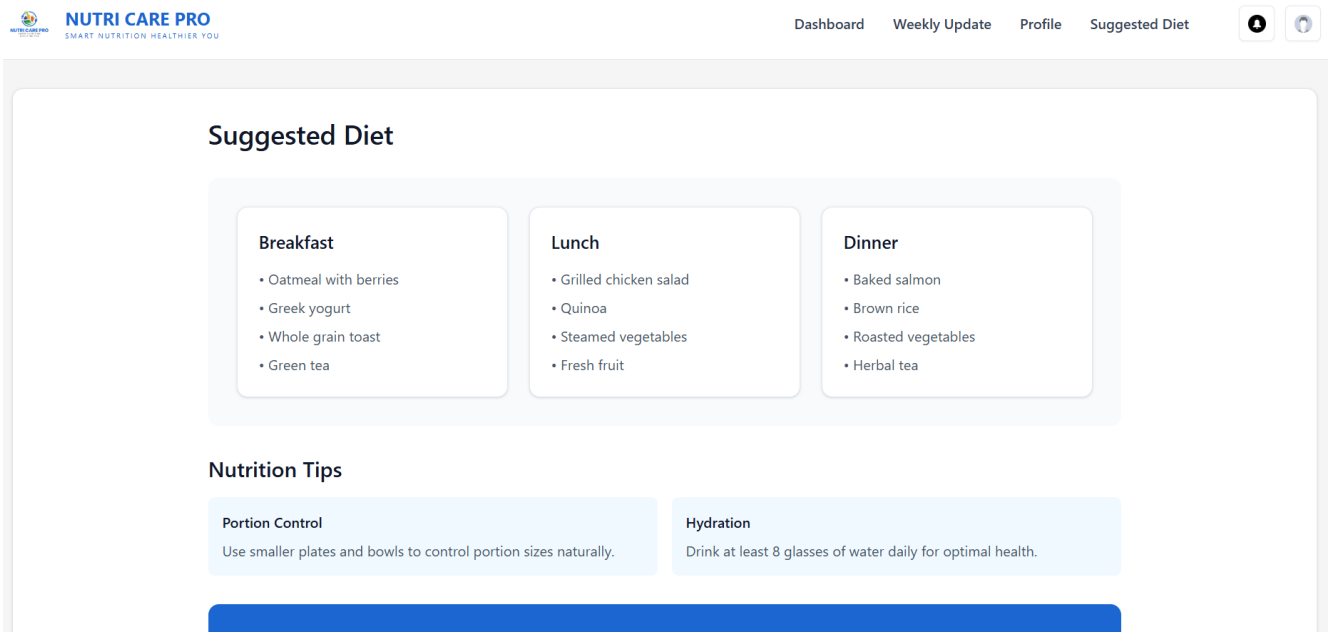
Hip (in)
42.5 ↓

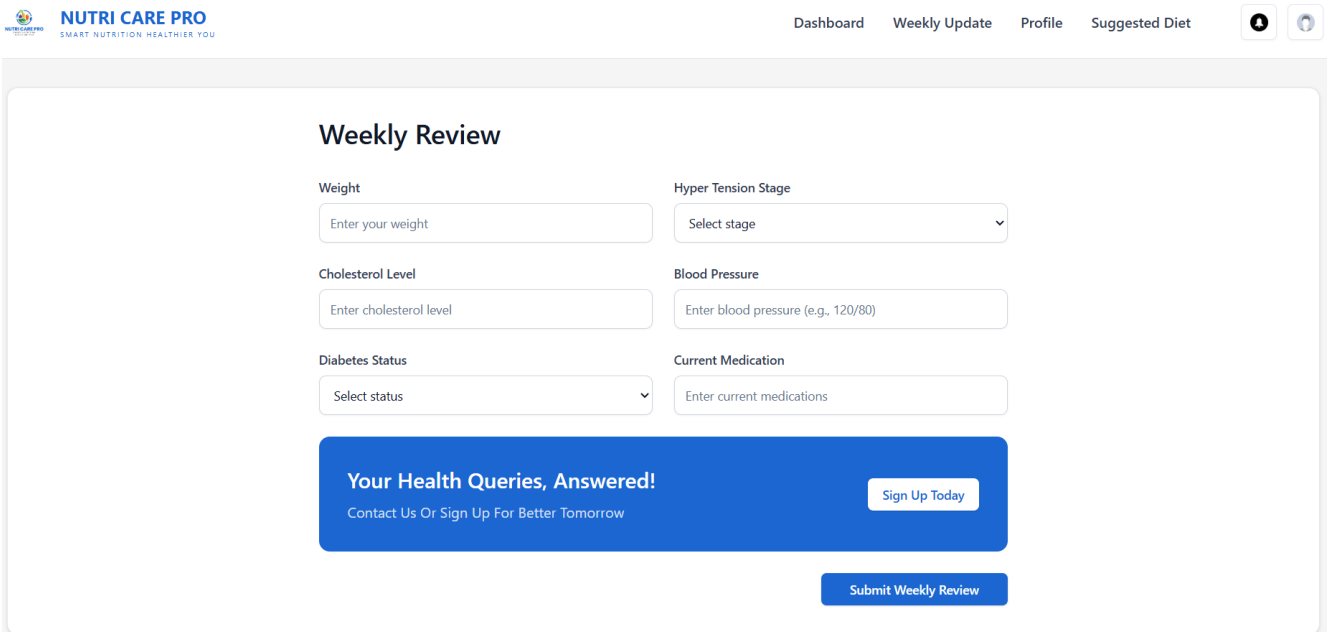
Body Shape

Inverted Triangle Body Shape




NUTRI CARE PRO
SMART NUTRITION HEALTHIER YOU
Dashboard
Weekly Update
Profile
Suggested Diet
Personal Information
First Name*
Gender
Current Medication
Date of Birth (DD-MM-YYYY)
DD-MM-YYYY
Weight
Cholesterol Level
Last Name*
Height
Diabetes Status
Age
Blood Pressure
e.g., 120/80
Salt Sensitivity
Smart Nutrition, Healthier You!
Contact Us Or Sign Up For Better Tomorrow
Sign Up Today


NUTRI CARE PRO
SMART NUTRITION HEALTHIER YOU
Dashboard
Weekly Update
Profile
Suggested Diet
Suggested Diet
Breakfast
• Oatmeal with berries
• Greek yogurt
• Whole grain toast
• Green tea
Lunch
• Grilled chicken salad
• Quinoa
• Steamed vegetables
• Fresh fruit
Dinner
• Baked salmon
• Brown rice
• Roasted vegetables
• Herbal tea
Nutrition Tips
Portion Control
Use smaller plates and bowls to control portion sizes naturally.
Hydration
Drink at least 8 glasses of water daily for optimal health.



5.6 Additional Information

- The system is designed to be **mobile-friendly** and **screen-reader accessible**.
- Local language support (Urdu) is planned for future releases.
- The app architecture allows **easy integration with wearable devices** like smartwatches and fitness bands.
- *UI decisions followed **Material Design** principles to improve user experience, especially for elderly users with chronic conditions.*

5.7 Other Design Details

- Research-backed models based on local nutrition data
- Plans for wearable device integration in the next release
- RESTful APIs built with modular endpoints

Chapter 6. Test Specification and Results

6.1 Test Case Specification

TC-1-User Authentication

Field	Description
Identifier	TC-1
Related Requirement	User Auth (FR-01), SRS §2.1
Short Description	Ensure a new user can register successfully.
Pre-condition	User not registered
Input Data	Valid name, email, and password
Detailed Steps	Open form → Fill → Submit
Expected Result	Account created, redirect to login.
Post-condition	New user saved in the database
Actual Result	☑ As expected
Test Case Result	**Pass**

6.2 TC-2 – Login Authentication

Field	Description
Identifier	TC-2
Related Requirement	Login Auth (FR-02), SRS §2.1
Short Description	Validate login with correct credentials.
Pre-condition	User exists and is verified.
Input Data	Email: test@mail.com, Password: correct123
Detailed Steps	Enter login form → Submit.
Expected Result	Redirect to dashboard
Post-condition	User session token created
Actual Result	☑ Redirect successful
Test Case Result	**Pass**

6.3 TC-3 – Diet Plan Generation

Field	Description
Identifier	TC-3
Related Requirement	Diet Generation (FR-05), SRS §2.3
Short Description	Ensure the diet is generated based on chronic conditions.
Pre-condition	The user is logged in with a profile.
Input Data	Age: 40, Weight: 85kg, Condition: Diabetes
Detailed Steps	Submit health form → Click "Generate Diet"
Expected Result	A condition-specific diet plan is shown.
Post-condition	Diet plan stored under the user account
Actual Result	☑ Correct diabetic-friendly plan shown
Test Case Result	**Pass**

6.4 TC-4 – Health Data Input

Field	Description
Identifier	TC-4
Related Requirement	Health Profile (FR-03), SRS §2.3
Short Description	Please verify that the user can input their health data successfully.
Pre-condition	User is logged in
Input Data	Age: 29, Weight: 72 kg, Condition: Hypertension
Detailed Steps	Open health form → Fill data → Click Save
Expected Result	Health data is saved in the user profile.
Post-condition	User record updated in the database.
Actual Result	☑ Health data stored and confirmed
Test Case Result	**Pass**

6.5 TC-5 – Health Progress Tracking

Field	Description
Identifier	TC-5
Related Requirement	Progress Monitoring (FR-07), SRS §2.4
Short Description	Ensure the user can view logged health metrics.
Pre-condition	Health logs already exist in the database.
Input Data	Date: May 10, Calories: 1800, Blood Sugar: 95
Detailed Steps	Submit log → View chart
Expected Result	Correct graph/chart displayed based on metrics
Post-condition	Progress shown in the visual chart
Actual Result	☑ Graph updated as expected
Test Case Result	**Pass**

6.6 Summary of Test Results

Module Name	Test Cases Run	Defects Found	Defects Fixed	Remaining Defects
Authentication Module	TC-1, TC-2	1	1	0
Diet Plan Generator	TC-3	2	2	0
Health Data Module	TC-4	1	1	0
Progress Tracker	TC-5	1	1	0
Complete System	**TC-1 to TC-5**	**5**	**5**	**0**

Chapter 7. Project Completion Status/Conclusion

The updated project Gantt chart below visualizes task timelines and current status.

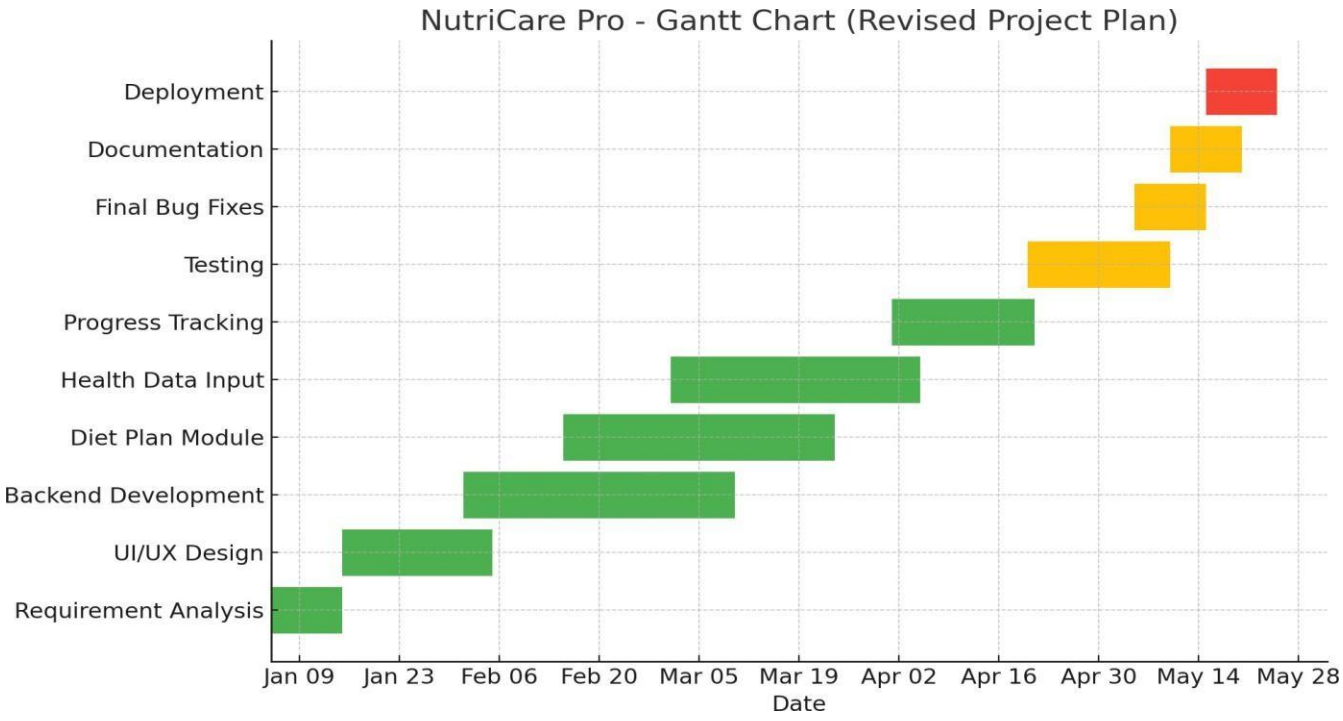


Table 7.1: Project Completion Status

Module Name	Status
User Authentication	☑ Complete
Health Data Input	☑ Complete
Diet Recommendation	☑ Complete
Health Progress Tracker	☑ Complete
Testing & Debugging	☑ Complete
UI Enhancements	☑ Complete
Deployment	❌ Not Yet Implemented
Final Documentation	🔄 Ongoing
Notification System	☑ Complete

Table 7.2: Objective(s)/Target(s) Status

Target/Objective	Status	Reason(s)
Objective 1: Generate personalized diet plans based on chronic diseases	☑ Completed	Successfully implemented ML-based diet generation tested via TC-3.
Objective 2: Support local food and supplement recommendations	☑ Completed	Diet plans incorporate Pakistani foods; an e-commerce module has been added.
Objective 3: Integrate real-time health tracking	☑ Completed	Health input forms and progress tracker implemented (see TC-4, TC-5)
Objective 4: Provide a user-friendly web	☑ Completed	Web app fully developed; mobile version not yet implemented
Number of Targets Completed	4	
Number of Targets Partially Completed	0	
Number of Targets Not Completed	0	

References

- Wahiduddin Khan (2016), *Nutrition and Health in South Asia*, ResearchGate
- MyFitnessPal Documentation
- Django Rest Framework – https://www.django-rest-framework.org
- MongoDB Docs – https://www.mongodb.com/docs
- TensorFlow Nutrition Models – https://www.tensorflow.org
- Stripe API – https://stripe.com/docs/api
- Google Fit API – https://developers.google.com/fit

Appendix A Glossary

- **BMI**: Body Mass Index
- **JWT**: JSON Web Token
- **RBAC**: Role-Based Access Control
- **ML**: Machine Learning
- **DRF**: Django Rest Framework

Appendix B IV & V Report

(Independent verification & validation)

IV & V Resource

Name Signature

S#	Defect Description	Origin Stage	Status	Fix Time	
				Hours	Minutes
1					
2					
3					
…					

Table B.2List of non-trivial defects: Deployment/Installation Guide

To deploy and install NutriCare Pro on your server:

8.1.1.1 Requirements:

- Python 3.10+
  - Node.js 18+
  - Docker & Docker Compose (for production)
  - PostgreSQL & MongoDB instances (locally or on MongoDB Atlas)

8.1.1.2 Local Development Setup:

Backend (Django):

1. Clone the repository:

   bash
   Copy
   Edit
   Git clone https://github.com/Moeez-Ur-Rehman/FYPNCP-Backend.git
2. Create a virtual environment:

bash
Copy
Edit
python -m venv venv source
venv/bin/activate

---

3. Install dependencies

   bash,
   Copy
   Edit
   , and pip install -r requirements.txt

4. Set up .envfile with DB credentials, Stripe key, etc.

5. Run server:

   bash
   Copy
   Edit
   python manage.py migrate python
   manage.py runserver

Frontend (React):

1. Navigate to the frontend directory:

   bash
   git clone https://github.com/Moeez-Ur-Rehman/FYP-NutriCarePro-Frontend.git

2. Install dependencies:

   bash
   CopyEdit
   npminstall

3. Run React dev server:

   bash
   CopyEdit
   npmstart

8.1.1.3 Production Deployment (Dockerized on AWS):

1. Update .envfiles for production settings.

2. Run:

   bash
   Copy
   Edit
   docker-compose up-- build -d

3. Ensure your EC2 server allows traffic on ports 80/443.

4. Assets (images, profile uploads) are served via AWS S3.

Appendix C User Manual

9.1.1.1 Getting Started:

1. Visit the app in your browser (e.g., http://localhost:3000).
2. Sign up with your email and password.
3. Log in to access your dashboard.

9.1.1.2 Using the App:

- **Step 1: Fill in Health Info**
  - Enter age, weight, dietary preference, and chronic condition.
  - Save your profile.
- **Step 2: Generate Diet**
  - Click "Generate Diet Plan".
  - Your personalized plan will appear with local food items.
- **Step 3: Track Health Progress**
  - Input daily health stats (calories, blood sugar).
  - View visual charts in the “Progress” tab.
- **Step 4: Edit Profile**
  - Navigate to the profile section to update your detail