

# Google Summer of Code 2025



## Open Healthcare Network

**Project Proposal: Diet Management in CARE**  
(Medium Project, 300hrs)

**Spandan Mishra**

**National Institute of Technology, Rourkela (NITR)**

**BTech (Bachelor of Technology), 4th Semester**

**Graduation Year: 2027**



**[Spandan-Mishra](#)**



**[Spandan Mishra](#)**

**[@spandanmishra1104@gmail.com](mailto:@spandanmishra1104@gmail.com)**

**+91 8260635598**

**Indian Standard Time (+5:30 GMT)**

# Index

<b>Project Proposal</b>	2-11
<b>Technical Skills and Relevant Experience</b>	12-13
<b>Implementation Timeline and Milestones</b>	13-14
<b>Summary About Me</b>	14-15
<b>Availability and Commitment</b>	15
<b>Contribution to OHC Repo</b>	16

# Project Proposal:

## Diet Management in CARE

---

### Project Overview

The Diet Management module aims to bridge the gap between clinical care and nutritional health within CARE's ecosystem. This module will empower patients to adjust their dietary plans, while also being able to monitor and track their eating habits. By integrating dietary data with existing patient records, CARE will unify medical and lifestyle interventions, addressing critical use cases such as:

- Chronic disease management: Diabetic patients can correlate blood glucose levels with carbohydrate intake.
- Realtime Monitoring: Doctors can monitor eating habits of their patients.
- Preventive care: Patients will be able to keep a track of their macros and calorie intake.

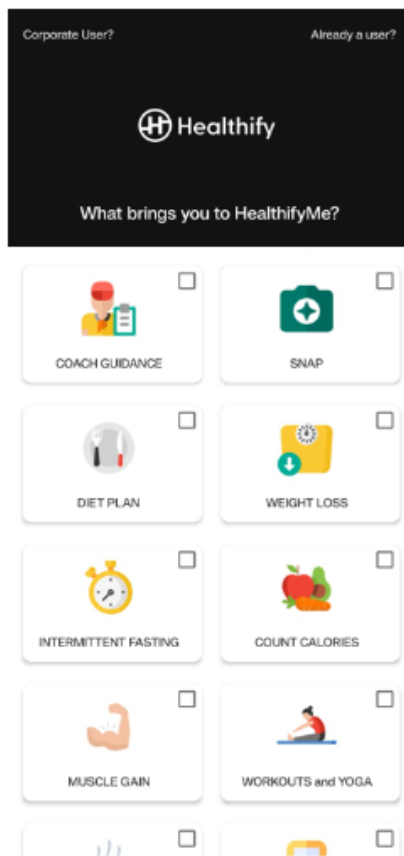
### Core Features & Technical Implementation

#### Meal Plan Integration

##### **Factors to consider:**

- Health conditions (eg. diabetes, hypertension) → Avoid foods which contribute to these medical conditions.
- Allergies → Some food like peanut butter, milk products cause allergies to people, and hence when specified shouldn't be added to their meal plan.
- Personal Preferences → Ranging from homemade, take-out food to taste preferences like spicy, minimalistic etc.
- Calorie goals: Many calorie tracking apps like [HealthifyMe](#) and [MyFitnessPal](#) let users choose their goals like maintain/lose/gain weight, or increasing nutrition content of their food, which further helps to personalize the meal plan.











Look of similar features on HealthifyMe:



Corporate User? Already a user?

Healthify

What brings you to HealthifyMe?

 COACH GUIDANCE	 SNAP
 DIET PLAN	 WEIGHT LOSS
 INTERMITTENT FASTING	 COUNT CALORIES
 MUSCLE GAIN	 WORKOUTS and YOGA
 	 

Prompting users for their goals and designing meal plans accordingly

-----

Any Medical Condition we should be aware of?

This info will help us guide you to your fitness goals safely and quickly.

☒ None

<input type="radio"/> Diabetes	<input type="radio"/> Pre-Diabetes	<input type="radio"/> Cholesterol
<input type="radio"/> Hypertension	<input type="radio"/> PCOS	<input type="radio"/> Thyroid
<input type="radio"/> Physical Injury	<input type="radio"/> Excessive stress/anxiety	
<input type="radio"/> Sleep issues	<input type="radio"/> Depression	<input type="radio"/> Anger issues
<input type="radio"/> Loneliness	<input type="radio"/> Relationship stress	

Prompting user to select medical conditions from a list of choices to prevent adding food that cause this, in their meal plan

## Architecture:

### Frontend:

- For new users: Dynamic form builder which allows users to select their medical conditions and allergies. Slider for choosing calorie goals.
- For old users: Data will be fetched from their existing records.
- Dashboard showing progress of the user, by integrating the calorie tracking feature along with current meal plan.
- Template meal cards, with expandable details showing nutritional value and calorie content.
- Updating dietary requirements for the user.

### Backend:

- End-point for user to create personalized meal plan based on conditions & preferences.
- Creating daily meal plans based on the current preferences and requirements of the user.
- Using [USDA FoodData Central API](#) to populate nutritional data of food and to fetch food which best fits the current meal plan.

### Database:

- Storing the users medical conditions as well as preferences.
- Storing food intake of the user by getting data from the Calorie Intake model, which can be used for later use.

### Additional/Future Scope:

- Medication based plans like Diabetic, Pre-Diabetic or any other health complications.
- ML model which analyzes user's history and recommends meal plan according to it.

## Calorie Tracking

## End Users:

There will be two types of users:

- Patients: People who log their foods, view real-time calorie intake and track macros.
- Clinicians/Doctors: Monitor the patient's daily eating habits and suggest changes accordingly.

## Methods to log:

- Pre-built databases like [USDA FoodData Central API](#) (free) and [Edamam](#) (paid) can help retrieve calorie and macronutrient data about the food which the user logs.
- [Open Food Facts API](#) helps to retrieve data about food via UPC (Universal Product Codes), which is a barcode symbology used for tracking trade items in stores. Scanning can be done by integrating [react-qrcode](#) library.
- User can also log custom foods with manual inputs. They must mention the portion sizes and the calorie count present in the food. This is helpful for tracking calorie of food which is not currently present in the database.

## Portion Sizes:

Conversion of portion sizes like cups, plates, oz into grams is important for standardizing the amount of intake. A library like [convert-units](#) can be used for this purpose. User input data can be converted into grams based on the portion they have chosen (cup, plates, oz, etc.) This way the user doesn't have to worry about calculating the grams present in the food, and the backend will handle the conversion. After getting the amount of food in grams, we calculate the calories present in it using formula like:

```
Calories in food = (Calories per 100g / 100) * Grams of food user has entered
```

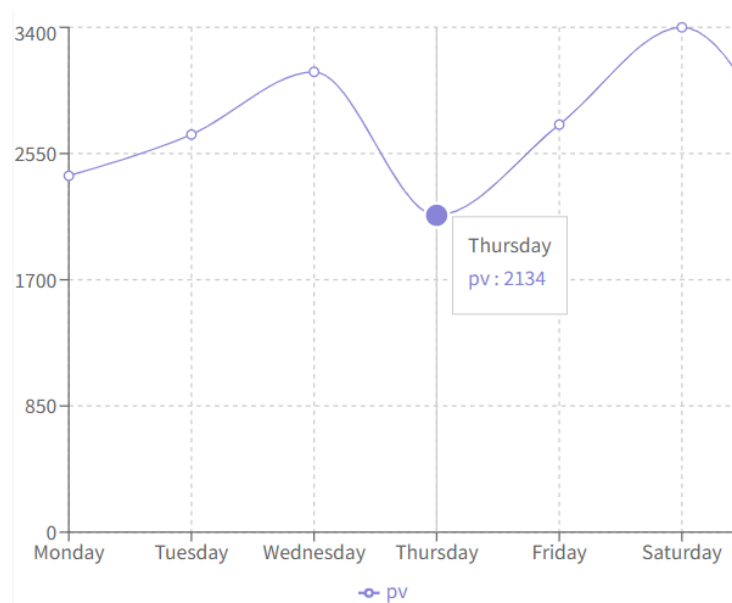
(Given, the standard value of Calories per 100g is known for the food being logged.)

## Calorie Tracking & Visualization:

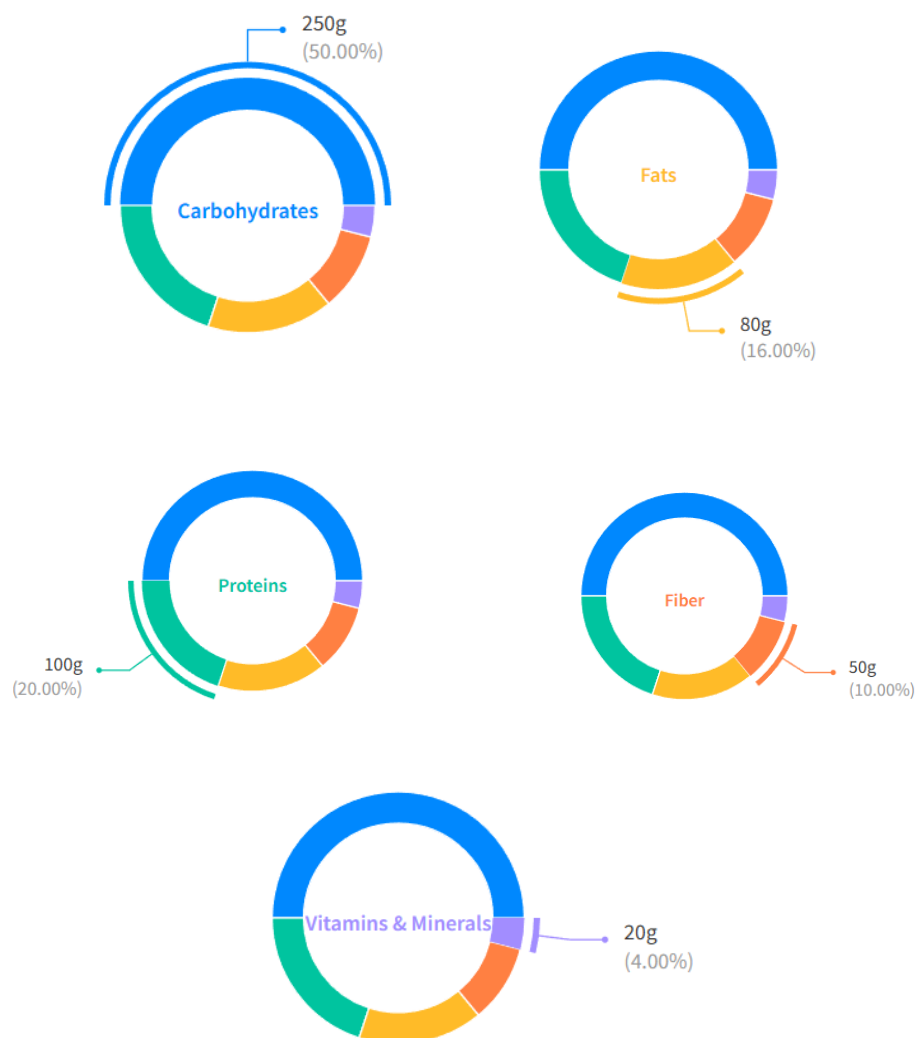
Calories logged by the user fill up a progress bar, based on the meal plan selected by the user (gain weight, lose weight etc.). Custom messages will be shown to the user based on the filling up of the progress bar.

For daily and weekly trends of the user's eating habits, charts can be shown on the frontend, displaying the distribution of macronutrients in their food. A library like [Recharts](#) can be used for visually analyzing the food intake. Pie charts and Bar charts can show the distribution of food of the user, while a line graph can show the calorie intake of the user on a weekly or monthly basis.

Example Image showing calorie intake over a week:



Pie Chart showing distribution of macronutrients in patient's meals:



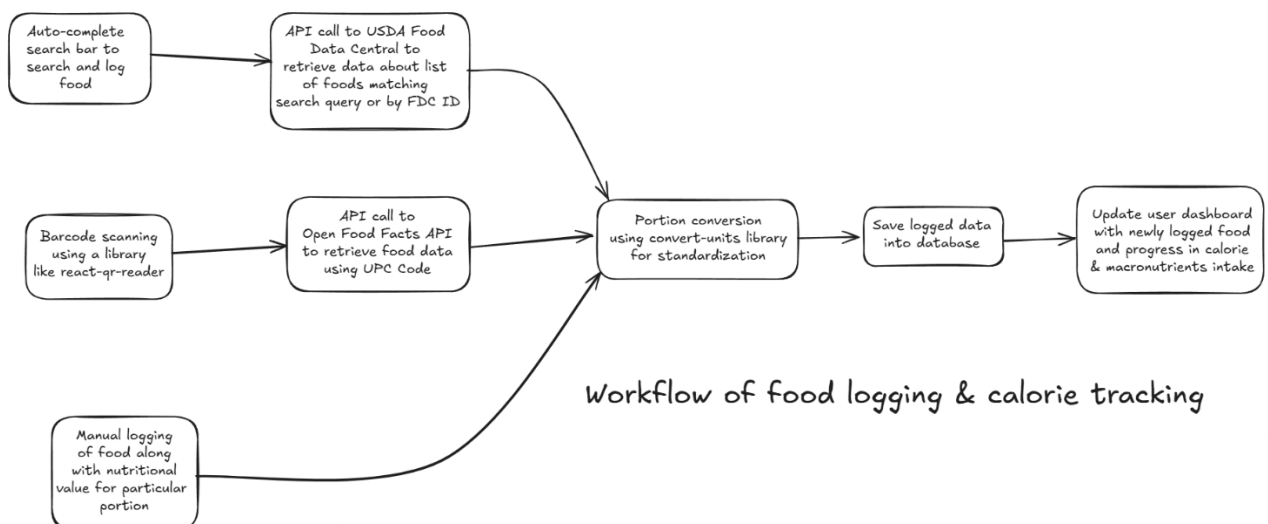
## Architecture:

### Frontend:

- Meal Logger - Auto-complete search bar with debounced API to reduce consistent database calls. Helps user to log food along with portion size to get calorie and macronutrient content.
- Barcode Scanner - Uses react-qr-reader to get the UPC codes and fetches data about the food from Open Food Facts API.
- convert-units library used to convert the portion size into standard units (grams).
- Daily Summary dashboard which shows the current calorie and macros progress of the user.
- Recharts library is used to visualize the data of the user and show it in terms of graphs and charts.

### Backend:

- Model to store the calorie log of the user, which will have fields like patient\_id (foreign key), food, quantity, logged\_at, meal\_type, source etc. (complete schema in Patient Record Integration Section).
- APIs to log food and to retrieve logs of a user.
- External Integration of APIs like USDA FoodData Central API to fetch standard nutrition data and Open Food Facts API to fetch nutritional content of food using UPC code.
- Caching frequently tracked foods in PostgreSQL Database to reduce calls to USDA API (allows 1000 calls/day).
- Fuzzy search to handle the typos made by users.





## Nutritional Recommendations

### User Information:

The Nutritional Recommendation feature in the Diet Management module in CARE is a data driven personalized dietary guide. It will generate nutritional recommendations for the user based on their personal information, health data and dietary patterns.

The data for the user can be categorized into the following:

Data Category	Parameters Collected	Source
Demographic	Age, Weight, Height, Gender, Activity Level	Patient Profile Module
Clinical	Medical Conditions (Diabetes, CKD), Allergies	Patient Information, FHIR Condition Resource
Biometric	Lab Results (HbA1c, LDL, eGFR)	FHIR Observation Resources
Behavioural	Dietary Preferences (Vegan, Low-Sodium)	User Preferences Form
Historical	30-Day Food Logs, Adherence Trends	Calorie Tracking Module

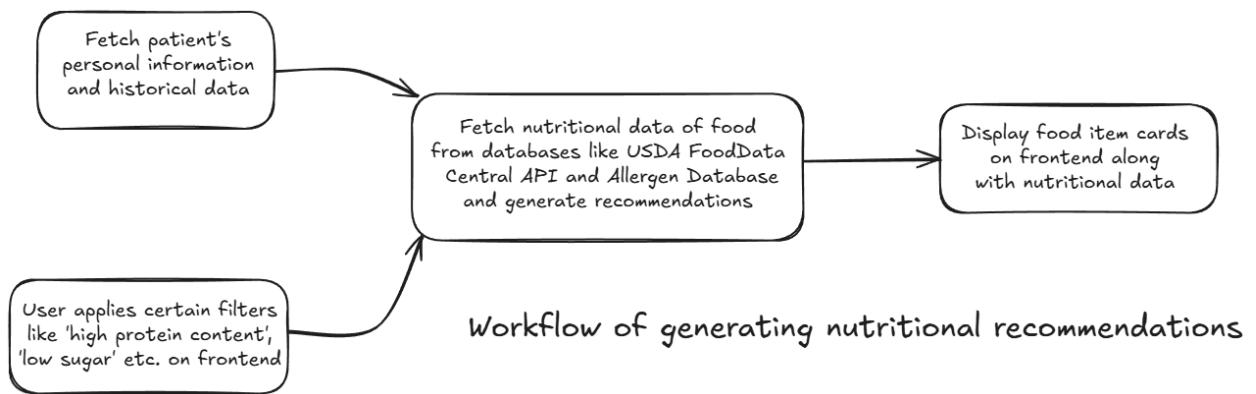
### Architecture:

Frontend:

- The calorie tracker dashboard is already present on the frontend. Below this a recommendation dashboard would be present, which shows the recommended food for the user and its nutritional values.
- Filters which allow users to toggle food based on their choices → Eg. Gluten Free, High Protein, Low Sugar.

Backend:

- Fetch data from the meal plan, calorie logs and patient information table and suggest food based on the differing factors.
- Retrieve nutrition data from databases like USDA FoodData Central API and Allergen Database (for excluding unsafe foods)



## Patient Record Integration

The dietary data of the patient can be converted into FHIR NutrionRecord resources for EHR interoperability. An additional table must be created in the database to store dietary data along with patient\_id as foreign key. This data can be fetched for a particular patient and will help in syncing the dietary data with overall patient record dashboard.

## Architecture:

Frontend:

- A patient dietary dashboard, which will consist of the complete meal plan component, calorie tracker, and recommendations tab.

Backend:

- Access control to allow only the patient and the specified clinicians to view the dietary data.
- APIs to fetch the meal plans, calorie logs and nutritional recommendations.

Database:

Store complete dietary information of the user in a Dietary Data table, which also takes data like current meal plan and the calorie logs stored in a separate table.

Schema for **Dietary Data** Table:

Column	Type	Description
id	UUID	Primary key
patient_id	UUID	Foreign key → patients.id
active_meal_plan_id	UUID	Foreign key → meal_plans.id (optional; if meal plans are stored)
recommendations	JSONB	Structured nutrient goals (e.g., {"protein": 60, "carbs": 150})
daily_summary	JSONB	Snapshot of daily intake (e.g., {"calories": 1800, "carbs": 200})
created_at	TIMESTAMPTZ	Timestamp of record creation (default: NOW())
updated_at	TIMESTAMPTZ	Timestamp of last update (auto-updated)

### Schema for **Calorie Log** Table:

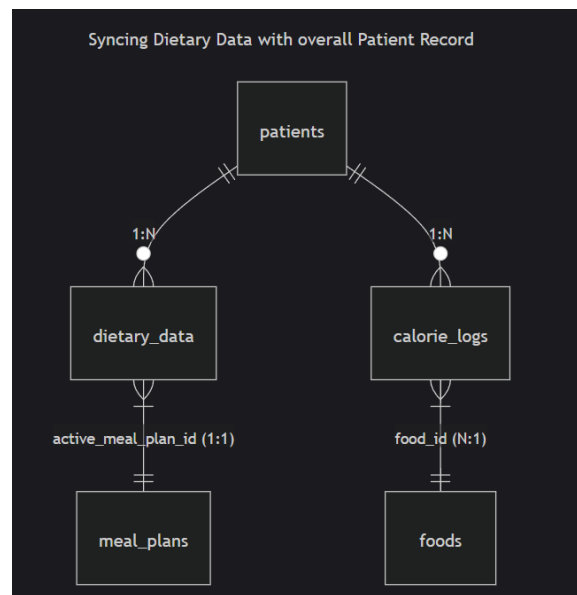
Column	Type	Description
id	UUID	Primary key (e.g., gen_random_uuid() in PostgreSQL)
patient_id	UUID	Foreign key → patients.id
food_id	UUID	Foreign key → food_id (linked to USDA FoodData Central or custom foods)
quantity	FLOAT	Quantity consumed (in <b>grams</b> )
logged_at	TIMESTAMPTZ	Timestamp of consumption (default: NOW())
meal_type	VARCHAR(20)	breakfast, lunch, dinner, snack
source	VARCHAR(20)	barcode, manual, usda_api
verified	BOOLEAN	TRUE if validated by a clinician

### Schema for **Meals Plan** Table:

This table is optional. In case we want to generate meal plans daily for the user, generating them using historical data and fetching data from USDA FoodData Central API would be a better approach.

Column	Type	Description
id	UUID	Primary key
patient_id	UUID	Foreign key → patients.id
conditions	JSONB	Medical constraints (e.g., ["diabetes", "hypertension"])
allergens	JSONB	Allergies (e.g., ["peanuts", "gluten"])
meals	JSONB	Structured meal schedule (e.g., {"breakfast": [food_ids], "lunch": [...]}))
is_active	BOOLEAN	TRUE if this is the current plan

### ER Diagram representing the overall relations:



## **List of 5 features you appreciate in the platform**

- **Role-Based Access Control:** The platform enforces strict role-based access, ensuring that doctors, nurses, admins, superadmins, and patients have permissions for their specific needs. This enhances security and privacy by restricting sensitive data access to authorized personnel. The implementation helps to prevent unauthorized modifications while allowing seamless interactions between different roles.
- **Fully Responsive and Modern UI:** A lot of effort has been put into making the UI clean, intuitive, and fully responsive across desktops, tablets, and mobile devices. The frontend efficiently adapts to different screen sizes without breaking the layout, ensuring a smooth user experience. This makes it accessible for medical professionals who may need to check patient data or reports on the go. The community is also very active in solving issues for the frontend.
- **Comprehensive Patient Reports & Encounter Tracking:** The platform allows generating detailed patient reports, including diagnostic results and treatment plans, giving doctors a structured view of patient history. The Encounter section is particularly powerful—it clusters critical patient details like allergies, diseases, symptoms, and treatments. The ability to generate Treatment Summaries and Discharge Summaries further enhances efficiency in patient management.
- **Centralized Staff Dashboard for Seamless Data Access:** Staff members get a unified dashboard where they can access everything related to them—facilities they manage, associations they are part of, and governance roles they hold. Instead of navigating through multiple sections, everything is available in one place, improving efficiency and decision-making. This reduces redundant navigation, enhances coordination, and ensures that staff can focus more on operational needs rather than searching for data.
- **Advanced Availability & Scheduling System:** The system allows staff to create precise availability slots, specifying time durations and the number of patients per slot. They can also define exceptions and unavailability periods, ensuring that appointment scheduling remains accurate and well-synced across the organization. This prevents double bookings, improves patient wait times, and streamlines coordination among doctors, nurses, and support staff.

# Technical Skills and Relevant Experience

## List of programming languages and technologies you're proficient

I have a solid MERN stack foundation, specializing in React, TypeScript, Next.js, Tailwind CSS, Framer Motion, and ShadCN for building modern, responsive UIs. On the backend, I'm proficient in Express.js, implementing JWT authentication, WebSockets, and optimized APIs. I have deep experience with MongoDB (data modeling, indexing, performance) and PostgreSQL for relational data.

Beyond JavaScript, I have a strong Python background, having built a full-stack project with Flask, and I'm currently advancing in Django.

## Description of relevant projects or contributions to open-source

### **Second Brain Application - [Repo](#)**

Developed Second Brain, a media management system that lets users store and organize various media such as tweets, YouTube videos, PDFs, and documents. I built a fully responsive frontend using React, Tailwind CSS, and TypeScript for added type safety. The backend is powered by Express.js, with a MongoDB database and JWT authentication. I also implemented a shareable link feature, allowing users to share their workspaces in a view-only mode, with the ability to enable or disable sharing in real time.

### **College Football - [Repo](#)**

Built an interactive football application using Flask and Jinja templating, with SQLite handling the database and HTML/CSS for the frontend. I worked with a large CSV dataset in Python to filter out and prepare relevant data for the application. The project features multiple interactive components where user inputs are processed to fetch and display the best fitting results, providing a polished and dynamic user experience.

## Collateral - [Repo](#)

Collateral is a real-time collaborative drawing application developed as a monorepo using Turborepo. The frontend is built with React, TypeScript, and Tailwind CSS, while the backend uses Node, Express, and WebSockets for real-time updates and PostgreSQL ORM is used to store data. In this project, users can join a room and draw together—any change made by one user is instantly visible to everyone in the room. This project helped me learn how to build scalable systems and manage interdependent packages within a unified repository.

## Open source contributions

I participated in Hacktoberfest '24, where I contributed to several projects. I worked on a Python-based repository, developing JWT-based authentication, learning a bit about FastAPI in the process. I also made contributions to a repository that uses CanvasAPI in JavaScript by introducing new themes for their cards. These experiences deepened my understanding of open source workflows and provided exposure to working on codebases built by other people and interacting with the maintainers to get more insight.

## Implementation Timeline and Milestones

Phases	Milestones	Duration / Dates
Community Bonding Period	<ul style="list-style-type: none"><li>Get familiar with the CARE codebase and architecture.</li><li>Engage with the community and mentors.</li><li>Discuss additional features and refine the project scope.</li></ul>	May 8 - June 1
Week 1	<ul style="list-style-type: none"><li>Design the UI/UX wireframes for the Diet Management module.</li><li>Gather and analyze user requirements for meal planning and calorie tracking.</li></ul>	June 2 - June 8
Week 2-3	<ul style="list-style-type: none"><li>Implement the basic structure for Meal Plan Integration.</li><li>Set up models and database schema for meal tracking.</li><li>Collect feedback on UI designs.</li></ul>	June 9 - June 22

<b>Week 4-6 and Evaluation 1</b>	<ul style="list-style-type: none"> <li>• Develop core functionality for meal logging and calorie tracking.</li> <li>• Start integrating dietary data with patient records.</li> <li>• Ensure UI consistency and accessibility. - Prepare deliverables for the midterm evaluation.</li> </ul>	June 23 - July 14
<b>Week 8-10</b>	<ul style="list-style-type: none"> <li>• Implement personalized nutritional recommendations.</li> <li>• Conduct testing and debugging on integrated features.</li> <li>• Ensure seamless synchronization with patient dashboards.</li> </ul>	July 15 - August 4
<b>Week 11-12</b>	<ul style="list-style-type: none"> <li>• Perform integration testing with patient records and other relevant modules.</li> <li>• Optimize database queries for efficiency.</li> <li>• Conduct performance benchmarking.</li> <li>• Fix potential bugs and refine user experience.</li> </ul>	August 5 - August 25
<b>Final Submission &amp; Evaluation</b>	<ul style="list-style-type: none"> <li>• Complete final project documentation.</li> <li>• Prepare the final report and submission.</li> <li>• Write a blog on the GSoC journey.</li> </ul>	August 26 - September 1

## Summary About Me

### Brief Introduction

Hey! I'm Spandan Mishra, a 2nd year BTech student at NIT Rourkela, interested in web development and open-source. My web development journey started seriously in July 2024 and since then I've been building with MERN, Next.js, TypeScript, WebSockets, MongoDB, PostgreSQL, and Python. Open source has been a whole new world for me. Initially I was overwhelmed when I started but I pushed through the challenges and considered mistakes as learning opportunities. This mindset has made me someone who loves to build in public, contribute to meaningful projects and be part of a community that fosters growth.

Apart from coding I enjoy working in collaborative environments, sharing knowledge and upskilling myself. Whether it's solving technical problems, improving my communication skills through discussions or contributing to projects that make an impact I thrive in spaces that encourage innovation and teamwork. Always up for a challenge I'm here to learn, build and grow!

## Your motivation for applying and what you aim to bring to the project

Contributing to the OHC repository excites me because of the profound impact that CARE has had on healthcare accessibility and management. Having used various healthcare apps in the past, I recognize the value of a well-integrated **Diet Management Module** in streamlining patient care. Nutrition plays a crucial role in recovery and treatment, and integrating this feature into CARE would enhance personalized care plans for patients, aligning with the platform's mission of improving healthcare outcomes. I am eager to bring my technical skills and experience to this module, ensuring it is user-friendly, scalable, and seamlessly fits within the existing system. Beyond this, I am deeply inspired by CARE's mission of using open-source innovation to solve real-world healthcare challenges. The impact CARE has had—from optimizing pandemic response to enabling TeleICU care in remote areas—is truly remarkable. Being part of such a driven and passionate team is a huge motivation for me, and I am excited to learn more about CARE's ecosystem, contribute meaningfully, and grow alongside this incredible community.

## **Availability & Commitment**

Starting in May, I will be on summer break from college until mid-July, which provides me with an excellent opportunity to focus on the project full-time. During this period, I can commit around **40 hours per week** without any conflicting engagements, ensuring that I have plenty of dedicated time to work on and contribute to the project. In the event of any urgent or emergency situations, I will proactively communicate with the team beforehand to ensure that project deadlines are met without any disruption. Even as my summer break ends and academic responsibilities resume, I'll remain fully committed to balancing both my studies and ongoing project work professionally and efficiently.

**Additionally, I do not have plans of submitting proposals to any other organization, whatsoever. I am 100% dedicated to contributing to the Open HealthCare Network.**



# Contribution to OHC Repo

PR Number	Bug Fix/Feature added
<a href="#">#11066</a>	Redesigned the dashboard for the user organization page
<a href="#">#11542</a>	Collapsible sections added in encounter overview
<a href="#">#11435</a>	Removed close button and fixed clicking outside issues in Duplicate Warning Message
<a href="#">#11417</a>	Fixed invalid nesting and DOM errors in Facility Location page
<a href="#">#11235</a>	Truncated questionnaire name
<a href="#">#11314</a>	Fixed content overflow in location page for smaller devices
<a href="#">#11094</a>	Fixed name overflow in user card
<a href="#">#11541</a>	Empty sections in Treatment Summary are hidden

All of my contributions to OHC can be found here -> [care fe](#)