### **User Stories за профили и автентикация**

1. Като потребител, искам да мога да се регистрирам, за да имам личен профил.
2. Като потребител, искам да мога да се логна с потребителско име и парола, за да получа достъп до моите данни.
3. Като потребител, искам да мога да се logout-на, за да приключа сесията си сигурно.

### **User Stories за добавяне на храна и калории**

1. Като логнат потребител, искам да мога да добавям какво съм ял, за да следя приема си.
2. Като логнат потребител, искам да въвеждам грамаж на храната, за да се изчисляват правилно калориите и макросите.
3. Като логнат потребител, искам автоматично да виждам колко калории, протеини, въглехидрати и мазнини съм приел въз основа на въведената храна.
4. Като логнат потребител, искам добавената храна да бъде записвана по дати, за да мога да виждам история на приема по дни.

### **User Stories за преглед на дневен прием**

1. Като логнат потребител, искам да виждам какво съм ял днес, за да следя текущия ми прием.
2. Като логнат потребител, искам да мога да преглеждам храната и макросите си от предишни дни, за да проследявам напредъка си.
3. Като логнат потребител, искам да виждам обобщение за деня – общ брой калории и макроси, за по-добър контрол върху диетата си.

### **User Stories за работа с храни**

1. Като логнат потребител, искам да мога да търся храни по име, за да ги намирам по-бързо.
2. Като логнат потребител, искам да мога да трия или редактирам храна, ако съм въвел грешка.

—————————————————————————————————————————

## **Основни Use Cases**

### 1. Регистрация на потребител

* Актьор: Нов потребител
* Описание: Потребителят въвежда потребителско име, имейл и парола, за да създаде профил.
* Резултат: Успешно създаден акаунт.

### **2. Вход в системата (Login)**

* Актьор: Регистриран потребител
* Описание: Въвежда потребителско име и парола, за да влезе в профила си.
* Резултат: Получава достъп до своите хранителни данни.

### **3. Преглед на дневен прием**

* Актьор: Логнат потребител
* Описание: Вижда списък с въведената храна за текущия ден и обобщена информация за калории, протеини, мазнини и въглехидрати.
* Резултат: Получава представа за дневния си прием.

### **4. Добавяне на храна**

* Актьор: Логнат потребител
* Описание: Избира храна от списък или въвежда име, добавя грамаж и системата автоматично изчислява макросите.
* Резултат: Храната се добавя към дневния прием.

### **5. Редактиране или изтриване на храна**

* Актьор: Логнат потребител
* Описание: Избира съществуваща храна от дневния си прием и я редактира или премахва.
* Резултат: Промените се отразяват в обобщената информация.

### **6. Преглед на история по дни**

* Актьор: Логнат потребител
* Описание: Избира дата от календар или списък, за да прегледа храната и макросите за избрания ден.
* Резултат: Вижда пълната информация за предишни дни.

### **7. Търсене на храна**

* Актьор: Логнат потребител
* Описание: Въвежда име на храна в търсачка и получава резултати от базата.
* Резултат: Може бързо да я избере и добави.

## **Клиент-Сървър (Client–Server) с REST API**

### Описание на архитектурата:

Приложението е разделено на клиентска част (frontend) и сървърна част (backend), които комуникират чрез HTTP заявки (REST API).

* Frontend – Реализиран с React.js, отговаря за потребителския интерфейс и взаимодействието с потребителя.
* Backend – Реализиран с Node.js, обработва логиката, заявките и достъпа до база данни.
* База данни – SQLite, съхранява потребители, храни, дневни записи и макроси.

#### 

* UI и логиката на данните са разделени → по-лесна поддръжка и развитие.
* REST API позволява интеграция с мобилни приложения в бъдеще без промяна на backend-а.
* Backend и frontend могат да се тестват поотделно и да се хостват независимо.
* Сървърът поема логиката, валидацията и защитата на данните (напр. JWT токени за автентикация).