# **Документация на проект: Уеб приложение за следене на хранене**

## **1. Въведение**

### **1.1 Цел на проекта**

Проектът представлява уеб базирано приложение, предназначено да помага на потребителите да следят какво ядат, колко калории и макронутриенти приемат. Потребителят създава акаунт, въвежда изядените храни, а приложението автоматично използва външно API – CalorieNinja – за извличане на съответната хранителна информация, която се записва в база данни. Основната цел е да се предостави удобно средство за анализ и следене на ежедневния прием на храна, което да улесни здравословния начин на живот.

### **1.2 Обосновка на избора на проект**

Темата за здравословното хранене е актуална, особено в контекста на съвременния начин на живот, който често включва ограничено време за приготвяне на храна и липса на баланс в хранителния режим. Съществуват различни мобилни и уеб приложения, които предлагат подобни услуги, но много от тях са платени, сложни или претрупани с функционалности. Настоящият проект цели да предостави опростено, интуитивно и безплатно решение, което да служи като добра основа за бъдещо развитие и добавяне на допълнителни възможности.

## **2. Изисквания към системата**

### **2.1 Функционални изисквания**

* Регистрация и вход чрез Firebase Authentication.
* Добавяне на хранителен запис от потребителя.
* Извличане на информация за храната чрез CalorieNinja API.
* Автоматично записване на хранителните стойности в база данни.
* Преглед на историята на храненията от регистриран потребител.
* Сигурност и достъп единствено до собствените данни.

### **2.2 Нефункционални изисквания**

* Сигурно съхранение и защита на лични данни.
* Възможност за разширяемост добавяне на нови функционалности.
* Интуитивен интерфейс, подходящ за мобилни и десктоп устройства.

### **2.3 Ограничения**

* Без поддръжка на офлайн режим.
* Зависимост от външно API за извличане на хранителни стойности.
* Не е включена възможност а редактиране на вече записани хранения.

## **3. Анализ на потребителските сценарии**

### **3.1 Сценарий 1: Създаване на профил**

Потребителят отваря началната страница и избира "Регистрация". След въвеждане на имейл и парола, Firebase Authentication създава акаунт и го удостоверява. След успешна регистрация потребителят се пренасочва към своето табло.

### **3.2 Сценарий 2: Добавяне на изядена храна**

Потребителят въвежда име на храна в текстово поле. Чрез backend сървъра се изпраща заявка към CalorieNinja API, като се използва въведената стойност. Получената информация (калории, въглехидрати, протеини, мазнини) се визуализира и след това се записва в базата данни, асоциирана с уникалния идентификатор на потребителя.

### **3.3 Сценарий 3: Преглед на храненията**

Потребителят има възможност да отвори секция „История“ или „Моите хранения“, където вижда списък с всички въведени храни, сортирани по дата. Данните се зареждат от сървъра чрез REST API и се визуализират в табличен или списъчен формат.

## **4. Сигурност**

### **4.1 Firebase Authentication**

* Удостоверяването се осъществява чрез имейл и парола.
* Всеки потребител има уникален uid.
* Token-ите се използват за удостоверяване при заявки към сървъра.

### **4.2 Безопасност на данните**

* Всички хранителни данни се свързват със съответния userId.
* Потребителите могат да достъпват само своите данни.

### **4.3 Предотвратяване на атаки**

* Валидация на входа.
* Token валидация от Firebase.

## **5. Срещнати трудности и решения**

### **5.1 Firebase интеграция**

**Проблем:** Някои части от Firebase Authentication SDK не работеха правилно при обновяване на страници.  
**Решение:** Използван е onAuthStateChanged, за да се запази сесията при рефреш. Това гарантира, че потребителят остава логнат при презареждане.

### **5.2 API ограничения**

**Проблем:** CalorieNinja има ограничен брой заявки на ден. При надвишаване се получава грешка 429 (Too Many Requests).

### **5.3 Достоверност на хранителните стойности**

**Проблем:** Някои храни се разпознават с ниска точност или връщат обобщени стойности.

### **5.4 Свързаност между бекенд и база**

**Проблем:** При грешна конфигурация на връзката с базата данни, данните не се записваха.  
 **Решение:** Добавен е отделен модул database.js с ясно управление на connection pool и логика за повторен опит при неуспех.

**5.5 Deployment - Проблем без решение**

## **6. Идеи за надграждане**

### **10.1 Добавяне на цели (goals)**

Позволяване на потребителя да въвежда дневни цели – калории, протеин и др. След това системата показва прогреса спрямо целта.

### **6.2 Графики и статистики**

* дневен прием на калории
* седмични макронутриенти
* тенденции с времето

### **6.3 Сканиране на баркод**

Може да се използва камерата на телефона, за да сканира хранителни продукти (чрез Open Food Facts API), и автоматично да попълни информацията.

### **6.4 Поддръжка на български език**

Интерфейсът и въвеждането на храни да поддържат кирилица. За това може да се използва българска база от хранителни стойности.

### **6.5 Web App**

Добавяне на offline режим и възможност за инсталиране като мобилно приложение.

## **7. Заключение**

Проектът представлява напълно функционално уеб приложение за следене на хранителен режим. Съчетавайки Firebase за удостоверяване, външно API за хранителни стойности и собствена база данни за съхранение, приложението демонстрира пълна разработка от край до край.

Проектът:

* Спазва добрите практики за разделяне на отговорност (frontend/backend).
* Интегрира външен доставчик на данни.
* Използва реална база данни с потребителска персонализация.
* Позволява надграждане в посока мобилност, визуализация и автоматизация.