Опис на задача: Expense Tracker API

Да се изгради **RESTful API** со Node.js и Express кое ќе овозможи **управување со лични трошоци**.
Секој корисник ќе може да се регистрира, најави и да ги следи своите трошоци според категорија и датум.

API-то треба да овозможи:

- **Регистрација на нов корисник**
    - Endpoint: `POST /auth/register`
    - Параметри: `email`, `password`
      Лозинката мора да биде **хаширана** (bcrypt)
    - Email адресата мора да биде уникатна
- **Најава на постоечки корисник**
    - Endpoint: `POST /auth/login`
    - Ако се точни → врати **JWT токен**
    - Сите понатамошни барања кон `/expenses` мора да имаат **Authorization header**:

Трошоци (Expenses)

API-то треба да овозможи CRUD операции за трошоци, но **само за најавениот корисник**.

Метод	Патека	Опис
POST	/expenses	Креира нов трошок
GET	/expenses	Ги враќа сите трошоци на корисникот
GET	/expenses/:id	Го враќа конкретниот трошок
PUT	/expenses/:id	Го ажурира трошокот
DELETE	/expenses/:id	Го брише трошокот

Филтрирање и извештаи

`GET /expenses?category=...`
→ враќа трошоци само од дадената категорија

`GET /expenses?from=2025-10-01&to=2025-10-15`
→ враќа трошоци во одреден временски период

`GET /expenses/summary`
→ враќа објект со вкупните трошоци по категорија и нивната сумаризација

Пример:

```
{
  "summary": {
    "храна": 1500,
    "транспорт": 600,
    "забава": 300
  },
  "total": 2400
}
```

Модел на податоци:

Табела: **users**

Поле	Тип	Опис
id	integer (PK)	уникатен идентификатор
email	string	уникатен email
password	string	bcrypt-хаширана лозинка

Табела: **expenses**

Поле	Тип	Опис
id	**integer (PK)**	**уникатен**
userId	**integer (FK → users.id)**	**на кој корисник му припаѓа**
title	**string**	**опис на трошок**
amount	**number**	**износ**
category	**string**	**категорија (пример: храна, транспорт, сметки)**
date	**date**	**датум на трошок**
createdAt	**date**	**автоматски timestamp**
updatedAt	**date**	**автоматски timestamp**

Технички барања:

- **Express.js за API рутирање**
- **SQLite или SQL база со Sequelize ORM или ORM библиотека по избор**
- **bcrypt за хаширање на лозинки**
- **jsonwebtoken (JWT) за автентикација**
- **error handler middleware за фаќање грешки и враќање на валиден JSON одговор**
- **Pagination со параметри `?page=` и `?limit=`**

Опционално :

- **Endpoint `/expenses/monthly`**

  → враќа сумирани трошоци по месец за тековната година

- **Dockerfile со кој може да се пушти целата апликација со `docker compose up`**

Expense Tracker — Frontend

Да се изгради апликација што им овозможува на корисниците:

- да се регистрираат и најават,
- да ги додаваат, прегледуваат, филтрираат и уредуваат своите трошоци,
- да добиваат визуелен преглед (графикони / табели) на своите месечни трошоци

Технологии (предлог, може и Next.js)

- **React** (со React Router и Axios)
  или **Angular** (со HttpClient и RouterModule)
- **State Management:**
  - React → Context API или Redux Toolkit
  - Angular → сервис + BehaviorSubject
- **UI Framework:**
  - Material UI (React) или Angular Material
- **Chart Library:**
  - Recharts (React) или ngx-charts (Angular)

Главни функционалности

Автентикација

Корисникот треба да може да се:

- **Регистрира**
  - Форма со email + password
  - По успешна регистрација → автоматски пренасочување кон Login
- **Најави**
  - При успешен login се зачувува JWT токен во `localStorage`
  - Токенот се додава во `Authorization` header за сите API повици
- **Одјава**
  - Избриши токен и врати корисник на Login страницата

Екрани:

- `/register`
- `/login`

Управување со трошоци

Корисникот може да:

- Ги прегледа сите трошоци во табела
- Додава нов трошок
- Ажурира постоечки
- Го брише трошокот

Полиња во формата:

- Title (string)
- Amount (number)
- Category (dropdown — пример: храна, транспорт, сметки, забава)
- Date (datepicker)

Екрани:

- `/expenses` → листа на трошоци
- `/expenses/new` → форма за додавање
- `/expenses/:id/edit` → форма за ажурирање

Филтрирање и пребарување

- филтер по категорија
- филтер по временски период (од — до)
- текстуално пребарување (пример: по наслов)

Овие вредности се праќаат како query параметри кон backend-от:

GET /expenses?category=храна&from=2025-10-01&to=2025-10-31

Извештаи и визуализација (бонус)

Посебна страница `/summary` која го прикажува **сумирањето по категорија** (податоците се добиваат од `GET /expenses/summary`).

Пример визуелен приказ:

- Pie chart или Bar chart со вкупните трошоци по категорија
- Прикажан вкупен износ под графиконот
- `/monthly` → графикон со вкупни трошоци по месец

Логика на frontend

- При додавање или уредување на трошок, полето `amount` мора да биде позитивен број
- Ако JWT токенот е невалиден или истечен → автоматски logout
- Сите грешки од backend треба да се прикажуваат како **toast/alert** пораки
- Формите треба да имаат **валидација** (нпр. задолжителни полиња)

Дополнителни поени (опционално)

- Pagination (на пример: 10 трошоци по страница)
- Dark / Light тема

Пример UX flow

1. Корисникот се најавува
2. Влегува на `/expenses` → гледа табела со сите трошоци
3. Клика „Add Expense“ → ја пополнува формата
4. Се враќа на листата → трошокот е додаден
5. Оди на `/summary` → гледа графикон со сумирање по категории