DEMO PROJECT FOR TRAINING

Project Initialization & Tooling

US-0.1 Create repository and base structure

As a developer, I want a clean project structure so that backend and frontend are separated and maintainable.

Scope

- Create Git repository
- Create folders `frontend/` and `backend/`
- Add root README

Acceptance Criteria

- Repo contains:
  - `/frontend`
  - `/backend`
  - `README.md`
- `README.md` includes project purpose and basic run commands

US-0.2 Setup Docker Compose for Postgres

As a developer, I want Postgres available locally using Docker to simplify environment setup.

Requirements

- docker-compose includes:
  - postgres
  - optional pgadmin

Acceptance Criteria

- `docker-compose up -d` starts Postgres successfully
- Postgres is accessible using configured credentials
- Environment variables are used (no hardcoded password)

Error Cases

- If DB container fails, logs should clearly show missing env vars

Backend Foundation (FastAPI + GraphQL)

US-1.1 Initialize FastAPI project

As a developer, I want a running FastAPI application to serve GraphQL APIs.

Requirements

- FastAPI setup
- Uvicorn dev server
- `/health` endpoint

Acceptance Criteria

- `GET /health` returns: **{ "status": "UP" }**
- Server starts with hot reload

US-1.2 Configure database connection (Postgres)

As a developer, I want the backend to connect to Postgres so that data can be persisted.

Requirements

- SQLAlchemy configuration

- Session management
- Config loaded from env variables

Acceptance Criteria

- Backend connects successfully using docker Postgres
- On startup, connection errors are logged clearly

Error Cases

- If DB credentials invalid, server returns meaningful log:
    - `"Database connection failed"`

Backend Data Model & Persistence

US-2.1 Create User database model

As a developer, I want a User table to store authentication credentials.

Fields

- id (UUID or SERIAL)
- username (unique, required)
- email (unique, required)
- password_hash (required)
- role (required: ADMIN / USER)
- created_at
- updated_at

Acceptance Criteria

- User table exists in DB
- username uniqueness is enforced
- email uniqueness is enforced

Validation Rules

- username: min 3 chars
- email: valid format
- password_hash: must not be null
- role: must be `ADMIN` or `USER`

US-2.2 Create Product database model

As a developer, I want a Product table to store product data.

Fields

- id
- name (required)
- description (optional)
- price (required, decimal)
- quantity (required, integer)
- created_at
- updated_at

Acceptance Criteria

- Product table exists in DB

Validation Rules

- name: required, min length 2
- price: required, must be >= 0
- quantity: required, must be >= 0

Backend Authentication (JWT)

US-3.1 Implement password hashing

As a developer, I want passwords stored securely.

Acceptance Criteria

- Password is hashed using bcrypt
- Password is never stored in plaintext

US-3.2 Implement JWT token generation

As a developer, I want JWT tokens generated so that frontend can authenticate.

JWT Payload

- userId
- username
- role
- exp

Acceptance Criteria

- JWT is signed with a secret key from env vars
- Token expiration is configurable

Error Cases

- If JWT secret is missing → backend fails with clear message

Backend GraphQL API (Strawberry GraphQL)

US-4.1 Expose GraphQL endpoint

As a developer, I want a GraphQL endpoint accessible so that the frontend can query/mutate data.

Acceptance Criteria

- `/graphql` endpoint is available
- GraphQL Playground is accessible

US-4.2 Implement GraphQL Auth Mutations (Register/Login)

As a user, I want to register and login so I can access the system.

GraphQL Mutations

- `register(username, email, password)`
- `login(username, password)`

Acceptance Criteria

- Register creates a new user with role USER by default
- Login returns:

  {
  "token": "jwt-token",
  "user": { "id": "...", "username": "...", "role": "USER" }
  }

Validation Rules

- username required
- email required
- password required (min 6)

Error Cases

- username exists → `"Username already exists"`
- email exists → `"Email already exists"`
- invalid login → `"Invalid credentials"`

US-4.3 Implement GraphQL Query "me"

As a logged-in user, I want to retrieve my profile information.

Query

- `me()`

Acceptance Criteria

- Returns logged-in user info
- Requires JWT

Error Cases

- no token → `"Unauthorized"`

Backend Product CRUD (GraphQL)

US-5.1 Query list of products

As a user, I want to list products so I can manage inventory.

Query

- `products()`

Acceptance Criteria

- Returns all products ordered by created_at desc
- Requires authentication

Error Cases

- no token → `"Unauthorized"`

US-5.2 Query product by ID

As a user, I want to view product details.

Query

- `productById(id)`

Acceptance Criteria

- Returns product if exists

Error Cases

- product not found → `"Product not found"`
- no token → `"Unauthorized"`

US-5.3 Create product mutation

As a user, I want to create products.

Mutation

- `createProduct(input)`

Input Fields

- name
- description
- price
- quantity

Acceptance Criteria

- Product is saved in DB
- Returned object contains id

Validations

- name required, min 2 chars
- price >= 0
- quantity >= 0

Error Cases

- invalid fields → `"Validation error: <field>"`
- no token → `"Unauthorized"`

US-5.4 Update product mutation

As a user, I want to update product details.

Mutation

- `updateProduct(id, input)`

Acceptance Criteria

- Product updated successfully

Error Cases

- product not found → `"Product not found"`
- invalid input → `"Validation error"`
- no token → `"Unauthorized"`

US-5.5 Delete product mutation (ADMIN only)

As an admin, I want to delete products.

Mutation

- `deleteProduct(id)`

Acceptance Criteria

- Only role ADMIN can delete
- Product removed from DB

Error Cases

- no token → `"Unauthorized"`
- role USER → `"Forbidden"`
- product not found → `"Product not found"`

Frontend Foundation (Angular + Tailwind + Material)

US-6.1 Initialize Angular project

As a developer, I want a strict Angular project setup.

Acceptance Criteria

- Angular created with strict mode
- app builds and runs

US-6.2 Setup Tailwind CSS

As a developer, I want Tailwind enabled for styling.

Acceptance Criteria

- Tailwind configured correctly
- Tailwind classes work in templates

US-6.3 Setup Angular Material

As a developer, I want Angular Material configured.

Acceptance Criteria

- Material theme installed
- Toolbar + Button render correctly

Frontend GraphQL Integration

US-7.1 Setup Apollo Angular client

As a developer, I want Apollo configured to call backend GraphQL.

Acceptance Criteria

- GraphQL endpoint configured in environment
- Example query works (products or health)

Error Cases

- If backend unreachable → show `"Network error"`

US-7.2 Attach JWT token to GraphQL requests

As a developer, I want the token attached automatically.

Acceptance Criteria

- If token exists, it is included in Authorization header:
    - `Authorization: Bearer <token>`
- If no token, request is sent without header

Frontend Authentication Module

US-8.1 Create Login Page (Reactive Forms)

As a user, I want to login to access the app.

UI Fields

- username
- password

Acceptance Criteria

- Both fields required
- Submit button disabled if invalid
- On login success:
    - token saved in localStorage
    - redirect to `/products`

Error Cases

- invalid credentials → show snackbar `"Invalid credentials"`
- network error → `"Server unreachable"`

US-8.2 Implement Logout

As a user, I want to logout securely.

Acceptance Criteria

- Token removed from localStorage
- Redirect to `/login`

US-8.3 Implement AuthGuard

As a developer, I want to protect routes.

Protected Routes

- `/products`
- `/products/new`
- `/products/:id/edit`

Acceptance Criteria

- Unauthenticated user redirected to `/login`

Frontend Layout & Navigation

US-9.1 Create main layout with toolbar + sidenav

As a user, I want consistent navigation.

Acceptance Criteria

- Toolbar visible on all authenticated pages
- Menu contains:

    - Products
    - Theme switch
    - Language switch
    - Logout

Product CRUD Frontend (Reactive Forms)

US-10.1 Products List Page

As a user, I want to see products in a table.

Requirements

- Material table
- Columns:
    - Name
    - Price
    - Quantity
    - Actions

Acceptance Criteria

- Table loads products from GraphQL
- Loading indicator shown during fetch

Error Cases

- Unauthorized → redirect to login
- GraphQL error → snackbar `"Failed to load products"`

US-10.2 Create Product Page

As a user, I want to add new products.

Form Fields

- name (required)
- description (optional)
- price (required)
- quantity (required)

Acceptance Criteria

- Uses ReactiveForms
- Validations:
  - name required, min length 2
  - price >= 0
  - quantity >= 0
- On success:
  - snackbar `"Product created successfully"`
  - redirect to `/products`

Error Cases

- invalid form → show validation errors under fields
- server error → snackbar `"Create product failed"`

US-10.3 Edit Product Page

As a user, I want to update an existing product.

Acceptance Criteria

- Form pre-filled using `productById`
- On save:
  - update mutation called
  - snackbar `"Product updated successfully"`
  - redirect to `/products`

Error Cases

- product not found → show page message `"Product not found"`
- server error → `"Update failed"`

US-10.4 Delete Product Action

As a user, I want to delete products.

UI Requirements

- Delete button triggers Material dialog confirmation

Dialog content:

- Title: `"Confirm deletion"`
- Message: `"Are you sure you want to delete this product?"`
- Buttons: Cancel / Delete

Acceptance Criteria

- If confirmed → delete mutation executed
- On success → snackbar `"Product deleted"`
- Table refreshes

Error Cases

- Forbidden → snackbar `"You are not allowed to delete products"`

Theme Switch (Dark / Light)

US-11.1 Implement theme switching

As a user, I want to switch between dark and light mode.

Acceptance Criteria

- Toggle available in toolbar
- Theme applies globally
- Preference stored in localStorage
- Reload keeps same theme

Internationalization (ngx-translate)

US-12.1 Setup ngx-translate

As a developer, I want translations managed via JSON files.

Acceptance Criteria

- Translation files exist:
  - `en.json`
  - `fr.json`
- Default language is EN

US-12.2 Translate UI labels

As a user, I want UI translated.

Must translate

- Login page labels
- Products list headers
- Buttons (Create / Edit / Delete / Save / Cancel)
- Validation messages
- Snackbar messages
- Menu items

Acceptance Criteria

- Switching language updates UI instantly
- Language saved in localStorage

Frontend Testing (Jest)

US-13.1 Setup Jest

As a developer, I want Jest configured instead of Karma.

Acceptance Criteria

- `npm test` runs Jest successfully
- Example test passes

US-13.2 Unit test AuthService

As a developer, I want AuthService tested.

Test Cases

- should store token on login success
- should clear token on logout
- should return true for isLoggedIn if token exists

Acceptance Criteria

- Tests pass

US-13.3 Unit test ProductsService (GraphQL)

As a developer, I want product GraphQL calls tested.

Test Cases

- should call products query
- should call create mutation
- should call update mutation
- should call delete mutation

Acceptance Criteria

- Apollo mocked properly
- Tests pass

US-13.4 Component tests

As a developer, I want key components covered.

Minimum Components

- LoginComponent
- ProductFormComponent
- ProductsListComponent

Acceptance Criteria

- Form validation tested
- UI elements presence tested

Error Handling & UX Quality

US-14.1 Centralized GraphQL error handling

As a user, I want consistent error feedback.

Acceptance Criteria

- Unauthorized error triggers logout + redirect
- Forbidden error shows snackbar `"Access denied"`
- Network error shows snackbar `"Server unreachable"`

US-14.2 Add loading indicators

As a user, I want feedback while operations run.

Acceptance Criteria

- Spinner shown on:
    - login

        - product list loading
        - create/update/delete

Finalization & Documentation

US-15.1 Write complete README

As a developer, I want clear setup instructions.

Must include

- prerequisites
- docker commands
- backend run commands
- frontend run commands
- test commands
- sample user credentials

Acceptance Criteria

- New developer can run project from scratch using README only

US-15.2 Code quality cleanup

As a reviewer, I want the code clean and consistent.

Acceptance Criteria

- No unused imports
- Consistent naming
- No commented dead code
- Lint passes
- At least 5 meaningful commits

Definition of Done (Global DoD)

A story is DONE only if:

- Feature implemented
- UI functional and styled
- Validations done
- Error cases handled
- Unit tests added where required
- No console errors
- Code pushed and documented