ChatGPT

= SPEC-1: Secure, Scalable FastAPI-Based ToDo App :sectnums: :toc:

== Background

The goal of this project is to create a secure, modular, and scalable ToDo web application using FastAPI. It follows OWASP secure coding practices, and is structured for progressive, maintainable development. This system includes registration, login, RBAC, user-specific ToDos, admin role management, logout, testing, Docker deployment, and CI/CD support.

== Requirements

Must Have: - [x] Secure registration system with email/password and hashed credentials - [x] Login system with role-based access control (RBAC) - [x] User-specific ToDo creation, listing, and deletion - [x] Admin dashboard for role management - [x] Logout and session clearing - [x] Unit testing using `pytest` - [x] Dockerization and `.env` support - [x] GitHub Actions CI/CD

Should Have: - [ ] HTTPS setup guide (external Nginx + Certbot) - [ ] UI enhancements using Tailwind or Bootstrap

Could Have: - [ ] REST API for external clients - [ ] JWT-based auth as an alternative to signed cookies

== Method

=== Architecture Overview

[plantuml]

@startuml package "Web Frontend" { [Home Page] --> [Register Form] [Home Page] --> [Login Form] [Login Form] --> [Dashboard] }

package "FastAPI App" { [main.py] --> [routes/register.py] [main.py] --> [routes/login.py] [main.py] --> [routes/dashboard.py] [main.py] --> [routes/todos.py] [main.py] --> [routes/admin.py] [main.py] --> [routes/auth.py] }

[Dashboard] --> [ToDos View] [Admin Panel] --> [Users List] [Login Form] --> [Session Cookie] @enduml

---

=== Database Schema (SQLModel)

User Table - id: int (PK) - email: str (unique) - hashed_password: str - role: str (default: "user") - created_at: datetime

ToDo Table - id: int (PK) - title: str - description: str - completed: bool - created_at: datetime - user_id: int (FK -> User.id)

=== Security Features - Passwords hashed using `passlib[bcrypt]` - Signed cookies for session via `itsdangerous` - RBAC: Admin-only access to `/admin` - Form validation on both frontend and backend - Cookie flags: `secure=True`, `httponly=True`

=== Configuration - `.env` used for secret and DB connection - Dockerized using a multi-stage build - GitHub Actions for CI (lint, type-check, test, coverage)

== Implementation

1. Scaffold project structure with FastAPI, Jinja2 templates
2. Create landing page with navbar and branding
3. Add `/register` route with form validation and secure persistence
4. Add `/login` route, hashed credential check, signed cookie session
5. Implement `/dashboard` with user greeting
6. Add `/todos` (CRUD) linked to logged-in user ID
7. Create `/admin` for role management (admin only)
8. Add `/logout` to clear cookies
9. Write unit tests for auth and ToDos
10. Containerize with Dockerfile and `.env` support
11. Add GitHub Actions CI pipeline

== Milestones

1. Project scaffold and landing page
2. Secure registration module
3. Login with session and RBAC
4. User dashboard and ToDo CRUD
5. Admin panel with role management
6. Logout and UI updates
7. Unit test coverage and testability
8. Dockerization and `.env`
9. CI/CD with GitHub Actions

== Gathering Results

- Use `pytest --cov` to measure test coverage (target: >90%)
- Manually verify access control via browser tests
- Check login/logout flows across roles
- Monitor production containers for memory and CPU usage
- Ensure secure headers and cookies in browser dev tools