Timesheet Management Application

Overview

The Timesheet Management Application is a web-based system for tracking employee work hours. It allows users to **register, login, submit timesheets**, and **view records for today or any specific day**. The application is built with:

- **Backend:** Spring Boot 2 and Java 8
- **Frontend:** Angular 20
- **Database:** MySQL (or any JPA-supported DB)
- **Authentication:** JWT (JSON Web Token)
- **Security:** Spring Security for authentication and authorization

---

Features

- **User Management**
  - Registration
  - Login
  - Logout
- **Timesheet Management**
  - Submit login and logout times
  - Fetch timesheet for today
  - Fetch timesheet for a specific date
- **Security**
  - JWT-based authentication
  - Angular route guards to restrict access to authenticated users

---

Prerequisites

- **Java JDK 8+**
- **Maven 3+**
- **Node.js 22+**
- **Angular CLI 20**
- **MySQL database** (or any JPA-supported database)
- Optional: Postman for testing APIs

Backend Setup (Spring Boot)

1 - Configure database

Update `application.properties` or `application.yml`:

```
spring.datasource.url=jdbc:mysql://localhost:3306/timesheet_db
spring.datasource.username=root
spring.datasource.password=root
spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true
```

2 Build and run backend

```
mvn clean install
mvn spring-boot:run
```

3 API Endpoints

Endpoint	Method	Description
`/api/auth/register`	POST	Register a new user
`/api/auth/login`	POST	Login user and get JWT token
`/api/auth/logout`	POST	Logout user
`/api/timesheet/submit`	POST	Submit timesheet
`/api/timesheet/today`	GET	Fetch today's timesheet
`/api/timesheet/day`	GET	Fetch timesheet for specific date

Frontend Setup (Angular 20)

1. Navigate to frontend folder

```
cd timesheet-web
```

2. Install dependencies

```
npm install
```

3. Run Angular application

```
ng serve --open
```

The app should now be running at `http://localhost:4200`.

4. Environment Configuration

Update `environment.ts` with backend URL:

```
export const environment = {
  production: false,
  apiUrl: 'http://localhost:8080/api'
};
```

JWT & Authentication

- After login, the backend returns a **JWT token**.
- The frontend stores the token in **localStorage** (`jwt_token`).
- Angular **AuthGuard** ensures only authenticated users can access protected routes (like Dashboard).
- To logout, the token is removed from localStorage.

Notes

- **Validation:** Backend DTOs have validation annotations to ensure correct input.
- **Security:** Spring Security + JWT protects API endpoints.
- **Testing:** Backend includes unit tests using JUnit and Mockito.
- **SSR Warning:** Ensure `localStorage` is accessed only on the client side (not during SSR).

Author

Shawky Elrifai