FailFeed Social Media Service

A Spring Boot-based REST API for a social media platform. Users can create accounts, follow each other, and share posts with their followers.

Quick Start

What This Does

- User registration and profile management
- Follow/unfollow other users
- Create and view posts
- Get personalized feeds based on who you follow

Running the Service

```
# Build and run
./gradlew bootRun

# Or on Windows
gradlew.bat bootRun
```

The service starts on `http://localhost:8080`

API Endpoints

The service has four main areas:

User Management

```
POST   /users/create           # Create a new user
GET    /users                  # List all users
GET    /users/{id}            # Get specific user
GET    /users/{id}/followers  # Who's following this user
GET    /users/{id}/following  # Who this user follows
```

Following System

```
POST   /users/{followerId}/follow/{targetId}   # Follow someone
POST   /users/{followerId}/unfollow/{targetId} # Unfollow someone
```

Posts

```
POST   /posts/create           # Create a new post
GET    /posts                 # List all posts
GET    /posts/feed/{userId}   # Get user's personalized feed
GET    /users/{id}/posts      # Get posts by specific user
```

Usage Examples

User Operations

```
# Create a user
curl -X POST "http://localhost:8080/users/create?name=Ciarann"

# List all users
curl http://localhost:8080/users

# Get specific user
curl http://localhost:8080/users/1

# See who follows user 1
curl http://localhost:8080/users/1/followers

# See who user 1 follows
curl http://localhost:8080/users/1/following
```

Following

```
# User 1 follows user 2
curl -X POST "http://localhost:8080/users/1/follow/2"

# User 1 stops following user 2
curl -X POST "http://localhost:8080/users/1/unfollow/2"
```

Posts

```
# Create a post
curl -X POST "http://localhost:8080/posts/create?userId=1&message= Welcome to
Oopsss!"

# Check your feed (posts from people you follow
curl http://localhost:8080/posts/feed/1

# See all posts in the system
curl http://localhost:8080/posts
```

```
# Get posts from just one user
curl http://localhost:8080/users/2/posts
```

How It Works

Creating Users: Just send a name, get back a user ID. Simple as that.

Following: Users can follow each other, but can't follow themselves. The system prevents duplicate follows too.

Posts: Each post belongs to one user. Feed shows posts from people you follow, combined into one list.

Feed Generation: Your feed pulls posts from everyone you're following and displays them together.

Error Handling

You'll get standard HTTP status codes back:

- **404 Not Found**: User doesn't exist
- **400 Bad Request**: Something invalid in your request (like trying to follow yourself)
- **500 Internal Server Error**: Something went wrong on our end

Errors return JSON like:

```
{
  "error": "What went wrong",
  "message": "More details about the problem"
}
```

Project Structure

The code is organized into standard Spring Boot layers:

```
src/main/java/com/failfeed/service/
├── controller/          # REST endpoints - what the API exposes
├── service/            # Business logic - the actual work gets done here
├── repository/         # Database access - talking to the database
├── model/             # Database entities - how data is stored
├── dto/               # API responses - what we send back to clients
└── exception/         # Error handling - managing what goes wrong
```

What You Can Do Here

User Stuff

- Register with just a name

- Look up any user
- See follower/following relationships

Social Features

- Follow/unfollow people
- Can't follow yourself (obviously)
- No duplicate follows allowed
- Efficient queries for followers

Posts

- Write posts with messages
- See posts from people you follow
- View everyone's posts
- Filter posts by specific users

Database

By default, it uses H2 in-memory database:

- Database name: `failfeeddb`
- Tables get created automatically
- Data disappears when service restarts
- H2 console available for debugging at `/h2-console`

Settings

Edit `src/main/resources/application.properties` to change things:

```
# What port to run on
server.port=8080

# Database connection
spring.datasource.url=jdbc:h2:mem:failfeeddb
spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true
spring.h2.console.enabled=true
```

Getting Started for Development

You'll need:

- Java
- Gradle (wrapper comes with project)

Useful commands:

```
# Run tests
./gradlew test

# Build everything
./gradlew build

# Clean and rebuild
./gradlew clean build
```

Hosting This Service application to be used by the devices on other network

- **Local Tunnel** - Npm package
  Install Local tunnel using npm,
- `npm install -g localtunnel`.
- `lt --port 8080(*YOUR-PORT-NAME*) --subdomain myfailfeed
  For the 2nd command you get the link. A Url that is available across every devices. Enter the link in the browser, it will ask for the password, scroll down and you will get the password in the link, then paste it inside the password section.

Use this URL as the baseURL in your Frontend application to access the endpoints in this service application

Stack

- **Spring Boot 4.0.0** - The framework
- **Java 25** - Language version
- **Spring Data JPA** - Database access layer
- **H2 Database** - Default database (in-memory)
- **Gradle** - Build tool

Troubleshooting

Port 8080 busy?: Change `server.port` in application.properties or kill whatever's using port 8080.

Database issues?: H2 resets when app stops - that's normal. Check `/h2-console` for debugging.

Build problems?: Try deleting the `.gradle` folder and rebuilding.