📱 Ngelmak Thruline Core

The **Ngelmak Thruline Core** is the heart of the Ngelmak ecosystem, responsible for handling the **social media logic** of the platform.
It manages posts, comments, likes, and user interactions, and integrates with other microservices through the API Gateway.

⚙ Features

- 🗄 **Postgres Database** — Persistent storage for social media data
- 🔑 **Vault Integration** — Secure management of secrets (JWT keys, DB credentials)
- 📡 **Future-ready Messaging** — RabbitMQ & Kafka planned for event-driven communication
- 🐳 **Dockerized Deployment** — Easy containerization and portability

---

📦 Dependencies

- **Spring Boot** — Core framework
- **Postgres** — Relational database
- **Vault** — Secret management
- **RabbitMQ / Kafka** — (planned) message brokers for async communication

---

🔑 Vault Configuration (Short Doc)

Thruline Core uses **HashiCorp Vault** to fetch secrets (e.g., JWT signing key, DB credentials).
Here's a minimal configuration example:

```
cloud:
  vault:
    uri: http://localhost:8200
    authentication: approle
    app-role:
      role-id: ${VAULT_ROLE_ID}
      secret-id: ${VAULT_SECRET_ID}
    kv:
      enabled: true
      backend: secret
      default-context: thruline-core
```

PROF

Steps:

1. Start Vault locally:

```
vault server -dev -dev-root-token-id=root
```

2. Enable KV secrets engine:

```
vault secrets enable -path=secret kv
```

3. Store secrets:

```
vault kv put secret/thruline-core db-username=postgres db-
password=supersecret jwt-secret=myjwtsecret
```

4. Configure `VAULT_ROLE_ID` and `VAULT_SECRET_ID` in your environment.

For details please check out Ngelmak-Vault repository.

🐳 Dockerfile

A sample `Dockerfile` for building the Thruline Core service:

```
# --- Build Stage ---
FROM maven:3.9.9-eclipse-temurin-21 AS builder

WORKDIR /app

# Copy Maven descriptor and download dependencies
COPY pom.xml .
RUN mvn dependency:go-offline -B

# Copy source code and build
COPY src ./src
RUN mvn clean package

# --- Runtime Stage ---
FROM openjdk:21-jdk AS runner

WORKDIR /app

# Copy built JAR from builder stage
COPY --from=builder ./app/target/*-SNAPSHOT.jar ./app.jar

# Expose service port
EXPOSE 4005
```

PROF

```
# Run the application
ENTRYPOINT ["java", "-jar", "app.jar"]
```

Build & Run

```
# Build the JAR with Maven
mvn clean package -DskipTests

# Build Docker image
docker build -t ngelmak-thruline-core .

# Run container
docker run -p 4005:4005 ngelmak-thruline-core
```

This way, your final image is **lightweight** (only contains JDK + compiled JAR) while the heavy Maven build tools stay in the builder stage.

🏃 Getting Started

Prerequisites

- Java 17+
- Maven 3.8+
- Postgres running locally (`localhost:5432`)
- Vault running locally (`localhost:8200`)

Run Locally

```
mvn spring-boot:run
```

PROF

📂 Project Structure

```
Ngelmak-Thruline-Core/
 ├── src/main/java/.../thruline
 │    ├── controller/        # REST endpoints
 │    ├── service/           # Business logic
 │    ├── repository/        # JPA repositories
 │    └── config/            # Vault, DB, messaging configs
 ├── src/main/resources/
 │    └── application.yml    # Configurations
 ├── Dockerfile              # Container build file
 └── pom.xml                 # Dependencies
```

🛠️ Roadmap

- ✅ Postgres integration
- ✅ Vault secret management
- 🔜 RabbitMQ integration for async messaging
- 🔜 Kafka integration for event streaming

📜 License

This project is licensed under the MIT License.
Feel free to use and adapt it for your own microservice architecture.

```
---

Would you like me to also add a **Mermaid diagram** showing how Thruline
Core interacts with Postgres, Vault, and (eventually) RabbitMQ/Kafka?
That could make the README more visual and easier to understand.
```