ZIRCUIT - Social Learning Platform

ZIRCUIT is a social learning platform that combines features from Twitter and LinkedIn to create a space for users to connect, share knowledge, and grow their skills.

Features

- **User Authentication**: Traditional login and OAuth2 (Google, Facebook, GitHub, LinkedIn)
- **Profile Management**: View and edit user profiles
- **Follow System**: Follow/unfollow users and view follower/following lists
- **Messaging**: Real-time chat between users
- **Notifications**: Real-time notifications for various user activities

Technology Stack

- **Backend**: Spring Boot 2.7
- **Database**: MySQL 8
- **Security**: JWT-based authentication with Spring Security
- **Real-time Communication**: WebSocket with STOMP
- **File Storage**: Local file storage with Spring Content
- **Caching**: Caffeine Cache

Prerequisites

- Java 17 or higher
- Maven 3.6 or higher
- MySQL 8.0 or higher

Setup Instructions

Database Setup

1. Create a MySQL database:

```sql
CREATE DATABASE zircuit_db CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci;
```

2. Configure database connection in `application.properties`:

properties

```
spring.datasource.url=jdbc:mysql://localhost:3306/zircuit_db?useSSL=false&serverTimezone=UTC&al
spring.datasource.username=your-username
spring.datasource.password=your-password
```

OAuth2 Configuration

Update the OAuth2 provider configurations in `application.properties`:

properties

```
spring.security.oauth2.client.registration.google.client-id=your-google-client-id
spring.security.oauth2.client.registration.google.client-secret=your-google-client-secret

spring.security.oauth2.client.registration.facebook.client-id=your-facebook-client-id
spring.security.oauth2.client.registration.facebook.client-secret=your-facebook-client-secret

spring.security.oauth2.client.registration.github.client-id=your-github-client-id
spring.security.oauth2.client.registration.github.client-secret=your-github-client-secret
```

JWT Configuration

Update the JWT configuration in `application.properties`:

properties

```
app.jwt.secret=your-jwt-secret-key
app.jwt.expiration=86400000
app.jwt.refresh-expiration=604800000
```

Building and Running the Application

1. Clone the repository:

bash

```
git clone https://github.com/your-username/zircuit.git
cd zircuit
```

2. Build the application:

bash

```
mvn clean package
```

3. Run the application:

bash

```
java -jar target/ZIRCUIT-0.0.1-SNAPSHOT.jar
```

Alternatively, you can run the application directly with Maven:

bash

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

The application will be accessible at: http://localhost:8080/api

API Documentation

Authentication API

- `POST /api/auth/register` - Register a new user
- `POST /api/auth/login` - Login with email and password
- `GET /api/auth/oauth2/{provider}` - Redirect to OAuth2 provider
- `POST /api/auth/oauth2/{provider}/callback` - Handle OAuth2 callback
- `POST /api/auth/refresh-token` - Refresh access token
- `GET /api/auth/verify-email` - Verify email address
- `POST /api/auth/forgot-password` - Request password reset
- `POST /api/auth/reset-password` - Reset password

User API

- `GET /api/users/{userID}/profile` - Get user profile
- `PUT /api/users/me/profile` - Update current user's profile
- `PUT /api/users/me/settings` - Update current user's settings
- `POST /api/users/me/avatar` - Upload profile avatar

Follow API

- `POST /api/users/{userID}/follow` - Follow a user
- `DELETE /api/users/{userID}/follow` - Unfollow a user
- `GET /api/users/{userID}/followers` - Get a user's followers
- `GET /api/users/{userID}/following` - Get users followed by a user

Message API

- `POST /api/messages` - Send a message
- `GET /api/messages/conversations` - Get conversations
- `GET /api/messages/{userID}` - Get conversation with a user
- `DELETE /api/messages/{messageID}` - Delete a message
- `PUT /api/messages/settings` - Update message settings
- `GET /api/messages/settings` - Get message settings

Notification API

- `GET /api/notifications` - Get notifications
- `PUT /api/notifications/{notificationID}/read` - Mark notification as read
- `PUT /api/notifications/read-all` - Mark all notifications as read
- `GET /api/notifications/unread-count` - Get unread notification count

WebSocket API

- `/ws` - WebSocket endpoint
- `/topic/messages/{userID}` - Topic for receiving messages
- `/topic/notifications/{userID}` - Topic for receiving notifications
- `/app/chat/{receiverId}` - Destination for sending messages

License