Use cases:

Authentication

- User creates a new account
- User logs in

Profile

- Get profile
- Update username
- Update email
- Update password

Friend request

- Add users as friends
- Block users
- Accept friend requests
- Get friend list
- Check pending friend request list

Post

- Creates post
- Get post by id
- Get user posts
- Update post
- Delete post
- Likes posts
- Get post comments
- Check feed
- Get user's liked posts
- Remove like

Comment

- Create comment
- Delete comment
- Update comment

Objects and Relationships

User:

```
@OneToMany(mappedBy = "user")
private List<Post> posts;

@OneToMany(mappedBy = "user")
private List<Comment> comments;
```

```
  @OneToMany(mappedBy = "user")
  private List<Like> likes;

  @ManyToMany
  @JoinTable(
        name = "friendRequests",
        joinColumns = @JoinColumn(name = "user_id"),
        inverseJoinColumns = @JoinColumn(name = "friend_id")
  )
  private List<Friend> friendRequests;
```

Post:

```
  @OneToMany(mappedBy = "post")
  private List<Like> likes;

  @Column(name = "posted_time")
  private LocalDateTime postedTime;

  @OneToMany(mappedBy = "post")
  private List<Comment> comments;

  @ManyToOne
  private User user;
```

Comment:

```
  @ManyToOne
  private Post post;

  @ManyToOne
  private User user;
```

Like:

```
  @ManyToOne
  @JoinColumn(name = "user_id")
  private User user;

  @ManyToOne
  @JoinColumn(name = "post_id")
  private Post post;
```

FriendRequest:

```
  @ManyToOne
  @JoinColumn(name = "user_id")
  private User user;

  @ManyToOne
  @JoinColumn(name = "friend_id")
  private User friendRequest;
```

User one to many -> Post
User one to many -> Like
User one to many -> Comment
User many to many -> FriendRequest

Post one to many -> Like
Post one to many -> Comment
Post many to one -> User

Comment many to one -> Post
Comment many to one -> User

Like many to one -> User
Like many to one -> Post

FriendRequest many to one -> User (sender)
FriendRequest many to one -> User (recipient)