The Facebook System

→ Facebook is a social media platform for users to connect with people and engage

→ Users can connect with other users by sending friend request or using the direct messaging

→ User can create profile where they create post to share with their friends.

→ Create pages regarding a topic to interest & group to form a community to similar people

→ FB generate a personalized feed for its users based on their friends, like pages, groups & the content they engage with to ensure the best experience.

Expectations from the interviewee

→ How are users able to search other user's profile?

→ Can users search other users using their phone numbers & email addresses

→ How are users able to add other users as friends?

→ How are users able to follow or unfollow other users without becoming a direct connection?

→ How are users able to create both groups & pages in addition to their user profile?

→ What are the criteria for joining a group? Can a user join a group if they are not friends with the creator of the group?

→ How will the system handle privacy lists with certain content to be displayed only a specified connections?

→ How do users set the privacy of groups? How does this change the

Way members & non-members interact with the group?

→ How will users be notified of pre-selected events?

→ How much control do users have in choosing what notification they can receive.

Observer Design Pattern → Can be used

Requirement Collection

R1: Users should be able to set the privacy of their profile page. They should also be able to create their profile page & add information such as work experience, education & place of living.

R2: Users of our system should be able to search for groups, pages & other users.

R3: Users should be able to write a new post & set its privacy

R4: Users should be able send as well as respond to the friend requests to other users. Users should be able to unfriend or block their users.

R5: Users should be able to follow other users without adding them to their friend list.

R6: Users should be able to like, share & comment on a post. They should also be able to like or comment on existing comment.

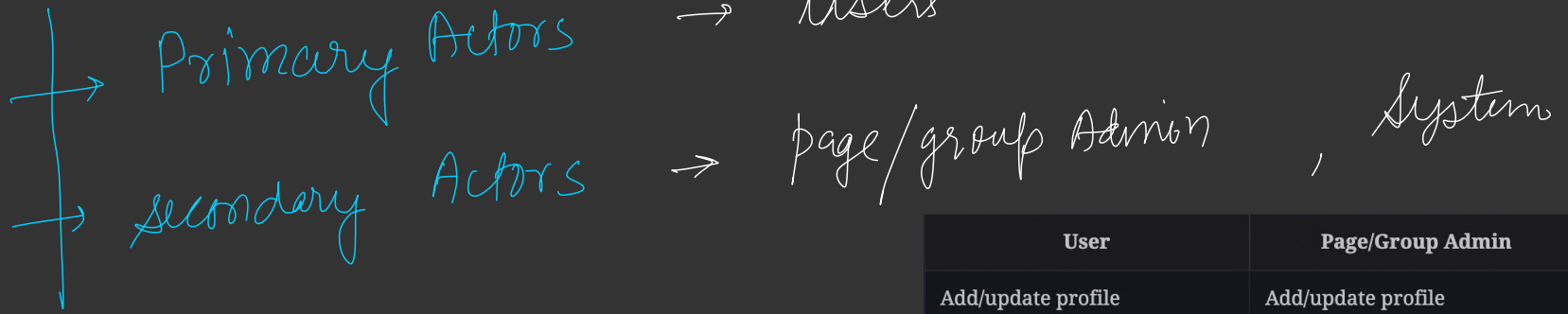
R7: The system should send the user a notification whenever there has been an interaction with them, such as receiving a message, a friend request, or a comment on their post.

R8: A user should be able to send messages & receive messages from other users.

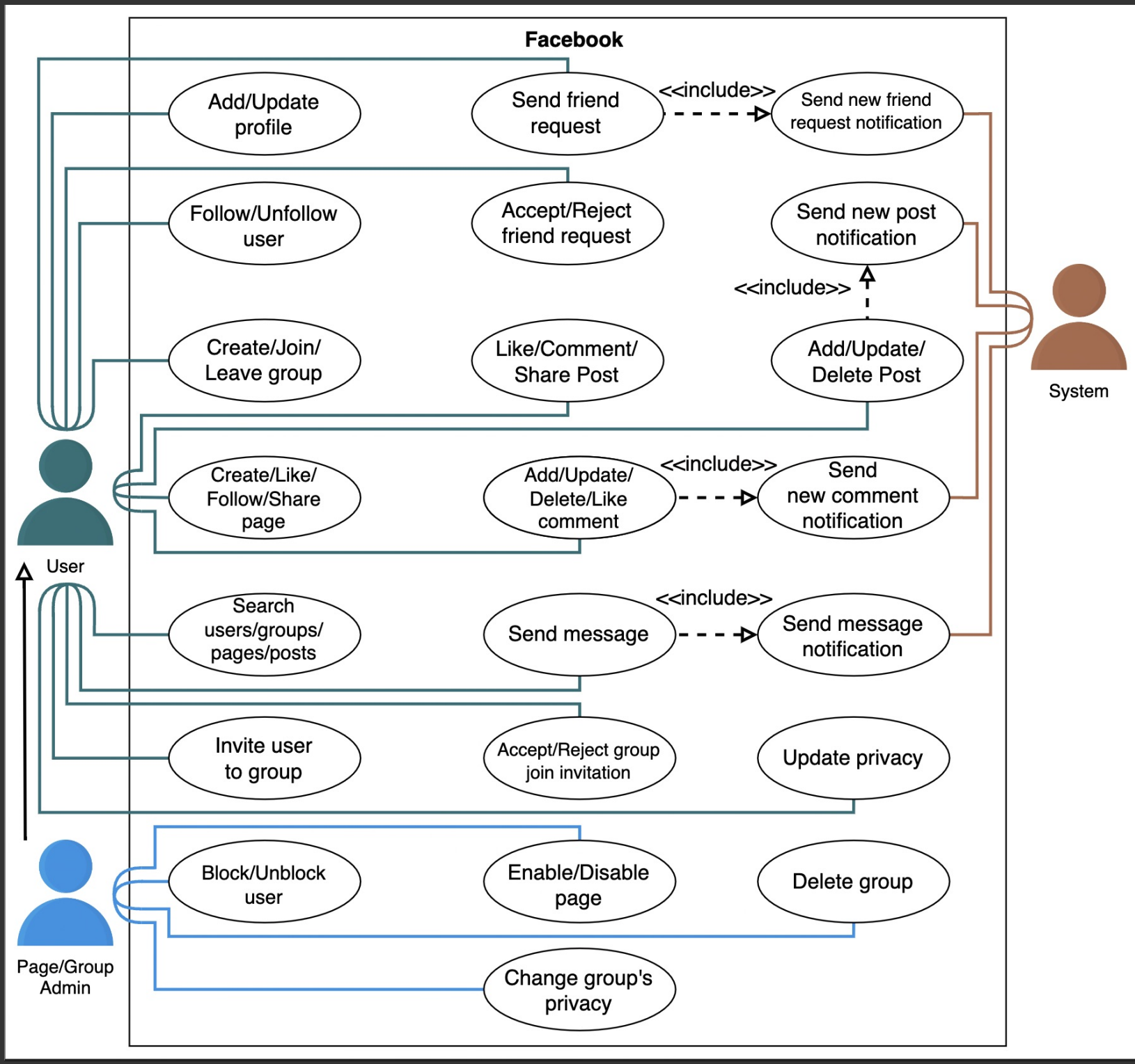
R9: Users should be able to follow existing pages & join existing groups. They should be able to unfollow or leave joined groups or followed pages.

R10: Users should be able to create their own groups & pages. Users can later set privacy or delete the groups or pages they own.

Actors



User	Page/Group Admin	System
Add/update profile	Add/update profile	Send new friend request notification
Follow/unfollow user	Follow/unfollow user	Send message notification
Send message	Send message	Send new post notification
Send friend request	Send friend request	Send comment notification
Create/like/follow/share page	Create/like/follow/share page	
Create/join/leave group	Create/join/leave group	
Invite user to group	Invite user to group	
Add/update/delete post	Add/update/delete post	
Add/update/delete/like comment	Add/update/delete/like comment	
Accept/reject friend request	Accept/reject friend request	
Update privacy	Update privacy	
Search users/groups/pages/posts	Search users/groups/pages/posts	
Accept/Reject friend request	Block/unblock user	
Accept/Reject group join invitation	Enable/disable page	
Like/Comment/Share Post	Delete group	
	Change a group's privacy	



Use Case Diagram

Class Diagram of FB

1. Account

Account

- status : AccountStatus
- accountID : string
- username : string
- password : string
- email : string

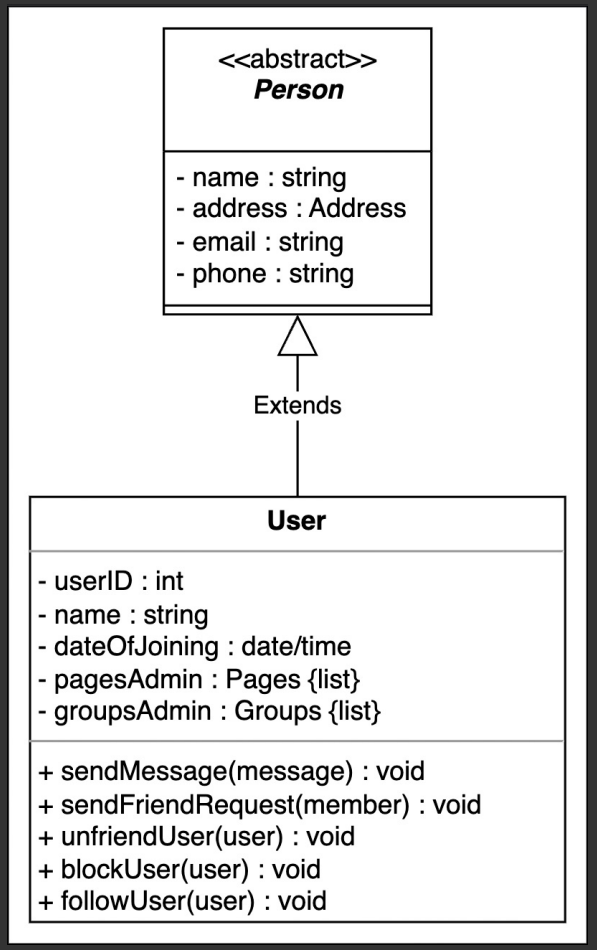
+ resetPassword() : bool

2. Admin

Admin

+ blockUser(user) : void
+ unBlockUser(user) : void
+ disablePage(page) : void
+ enablePage(page) : void
+ deleteGroup(group) : void
+ changeGroupPrivacy(group) : void

3. User



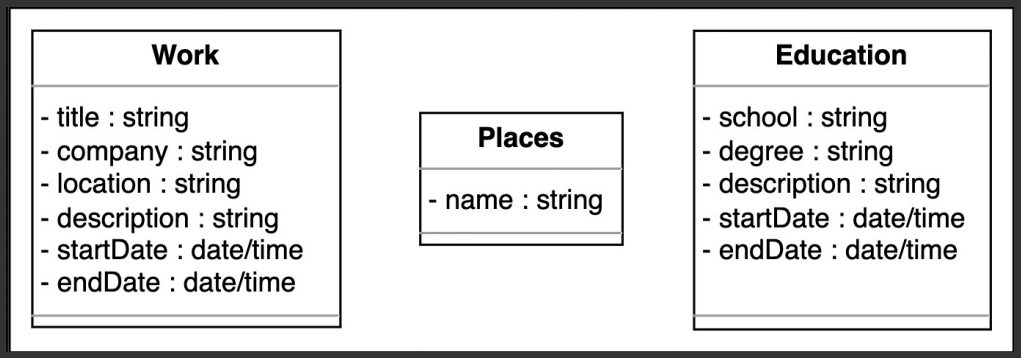
4. Profile

Profile

- profilePicture : byte {list}
- coverPhoto : byte {list}
- gender : string
- friends : Users {list}
- workExperience : work {list}
- educationInfo : Education {list}
- places : Place {list}

+ addWorkExperience(work) : bool
+ addEducation(education) : bool
+ addPlace(place) : bool
+ addProfilePicture(image) : bool
+ addCoverPhoto(image) : bool
+ addGender(gender) : bool

5. Work, Education & Places



6. Page

Page

- pageID : int
- name : string
- description : string
- likesCount : int

7. Group

Group

- groupID : int
- name : string
- description : string
- coverPhoto : byte[]
- totalUsers : int
- isPrivate : bool
- users : Users {list}

+ updateDescription(description) : void
+ addCoverPhoto(image) : void

8. Group functions

<<Interface>>
GroupFunctions

+ addUser(user) : void
+ deleteUser(user) : void
+ notifyUser(user) : bool

9. Post

Post

- postID : int
- text : string
- likesCount : int
- shareCount : int
- postOwner : User
- settings : PostPrivacySettings

+ changePostVisibility(post) : bool

10. Comment

Comment
- commentID : int - content : string - likesCount : int - commentOwner : User

11. FriendRequest

FriendRequest
- recipent : User - sender : User - status : friendRequestStatus - requestSent : date/time - requestStatusModified : date/time
+ acceptRequest() : bool + rejectRequest() : bool

12. Message

Message
- messageID : int - sender : User - recipients : User {list} - content : string - media : byte {list}
+ addRecipent(user {list}) : bool

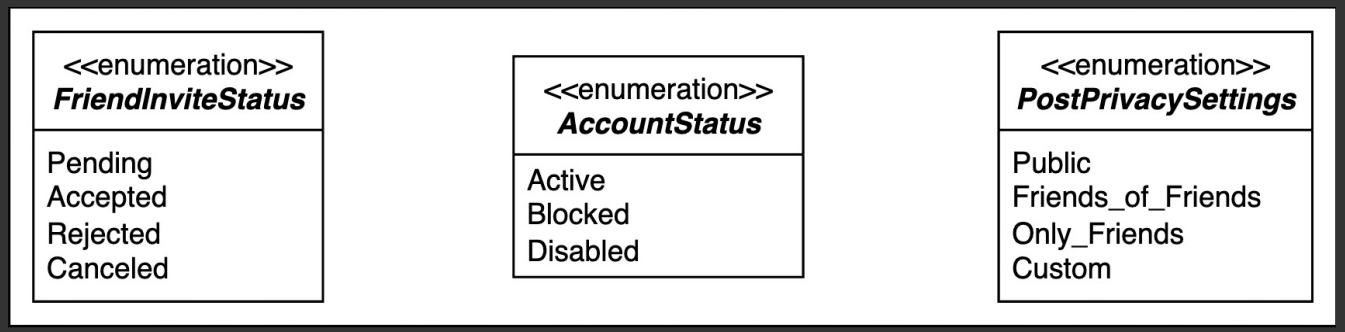
13. Profile privacy

ProfilePrivacy
+ changeFriendsListVisibility(profile) : void + lockProfile(profile) : void + lockProfilePicture(profile) : void

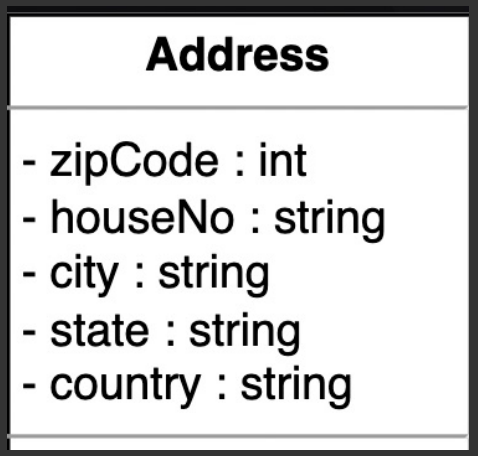
14. Notification

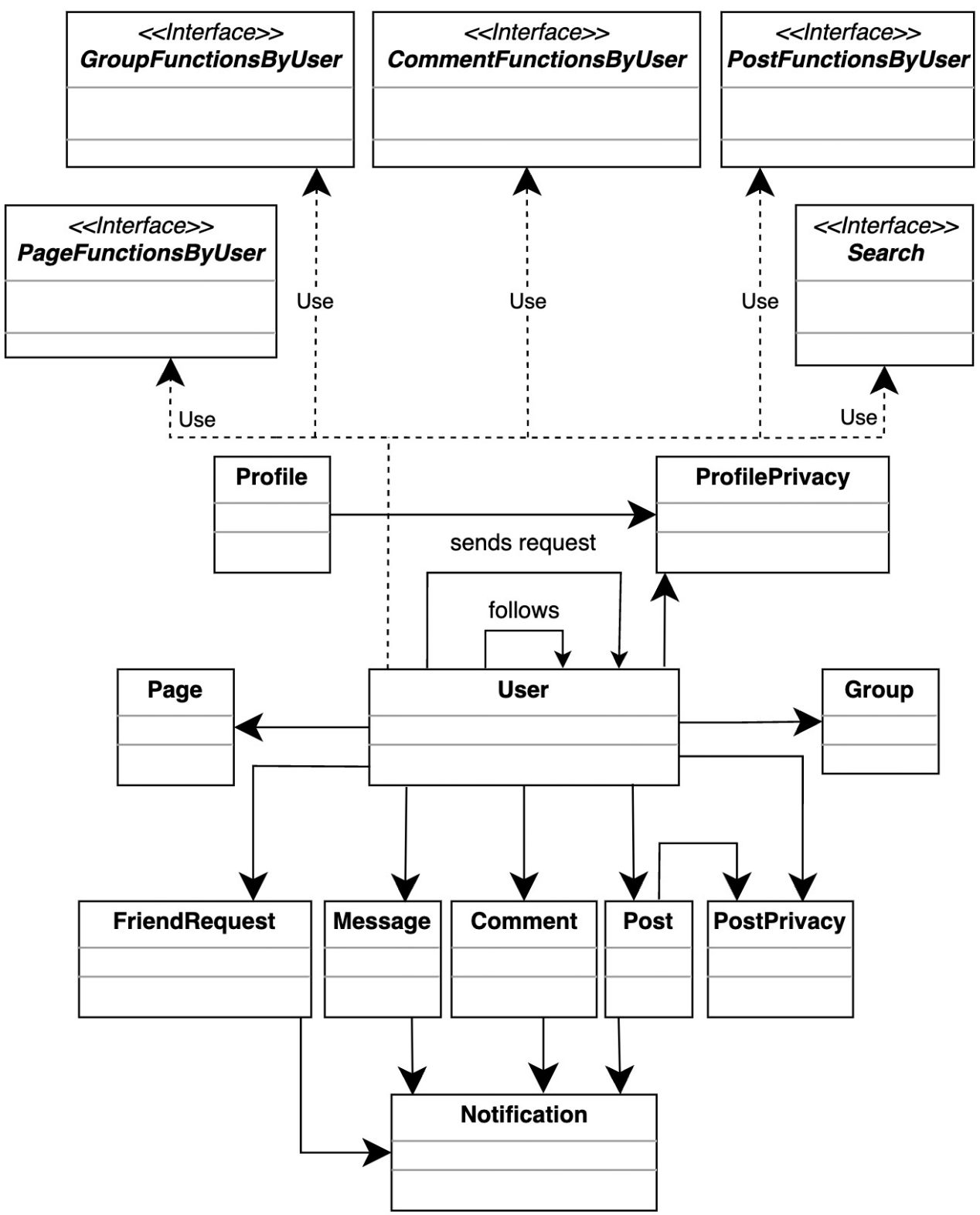
Notification
- notificationID : int - createdOn : date/time - content : string
+ sendNotification(account) : void

15. Enumerations

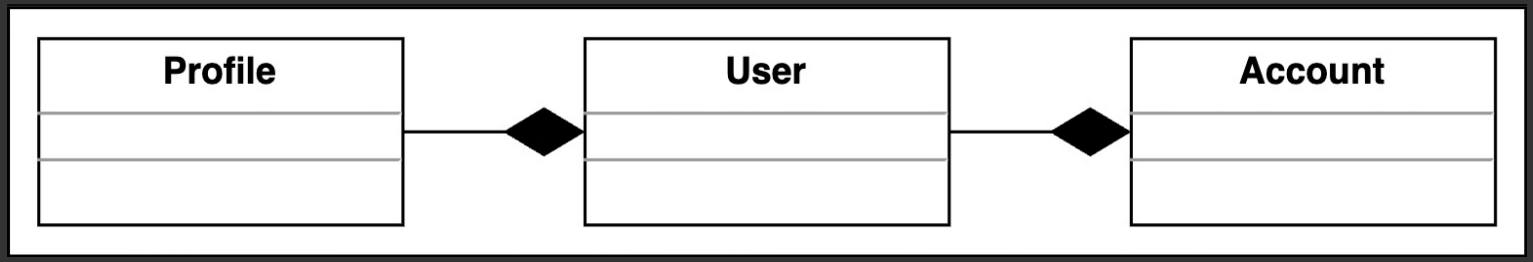


16. Address

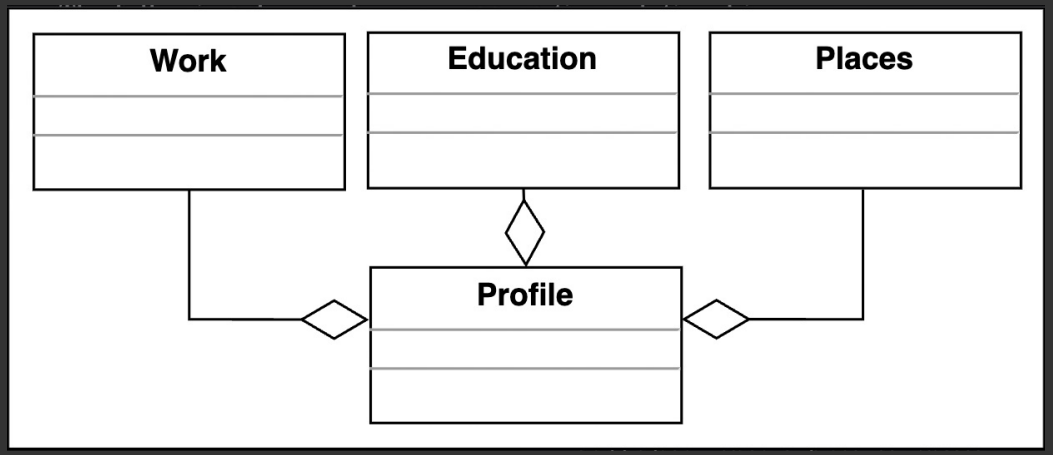




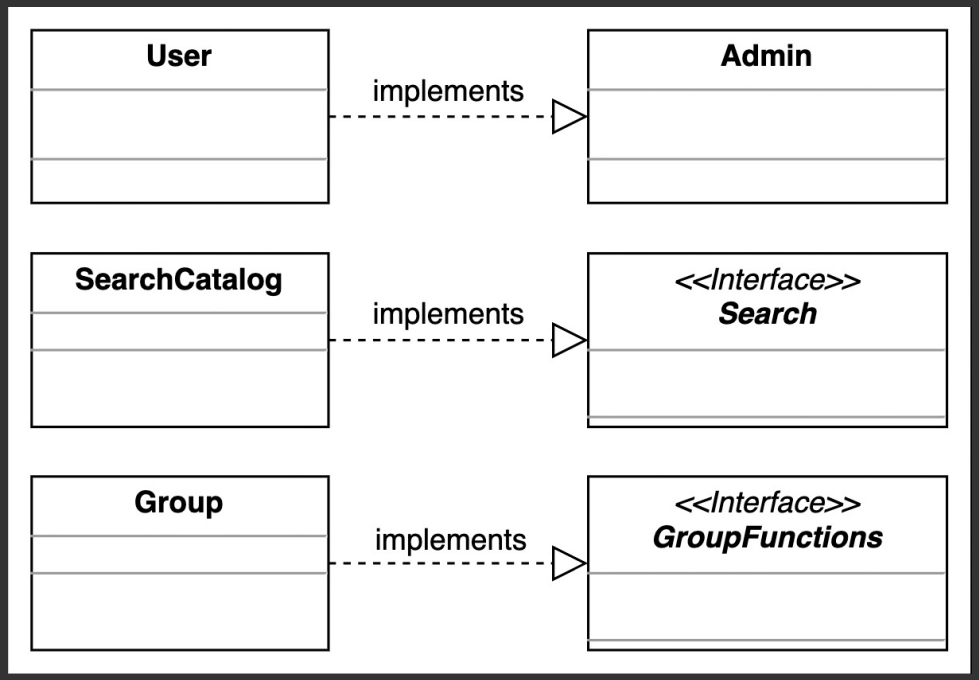
Association



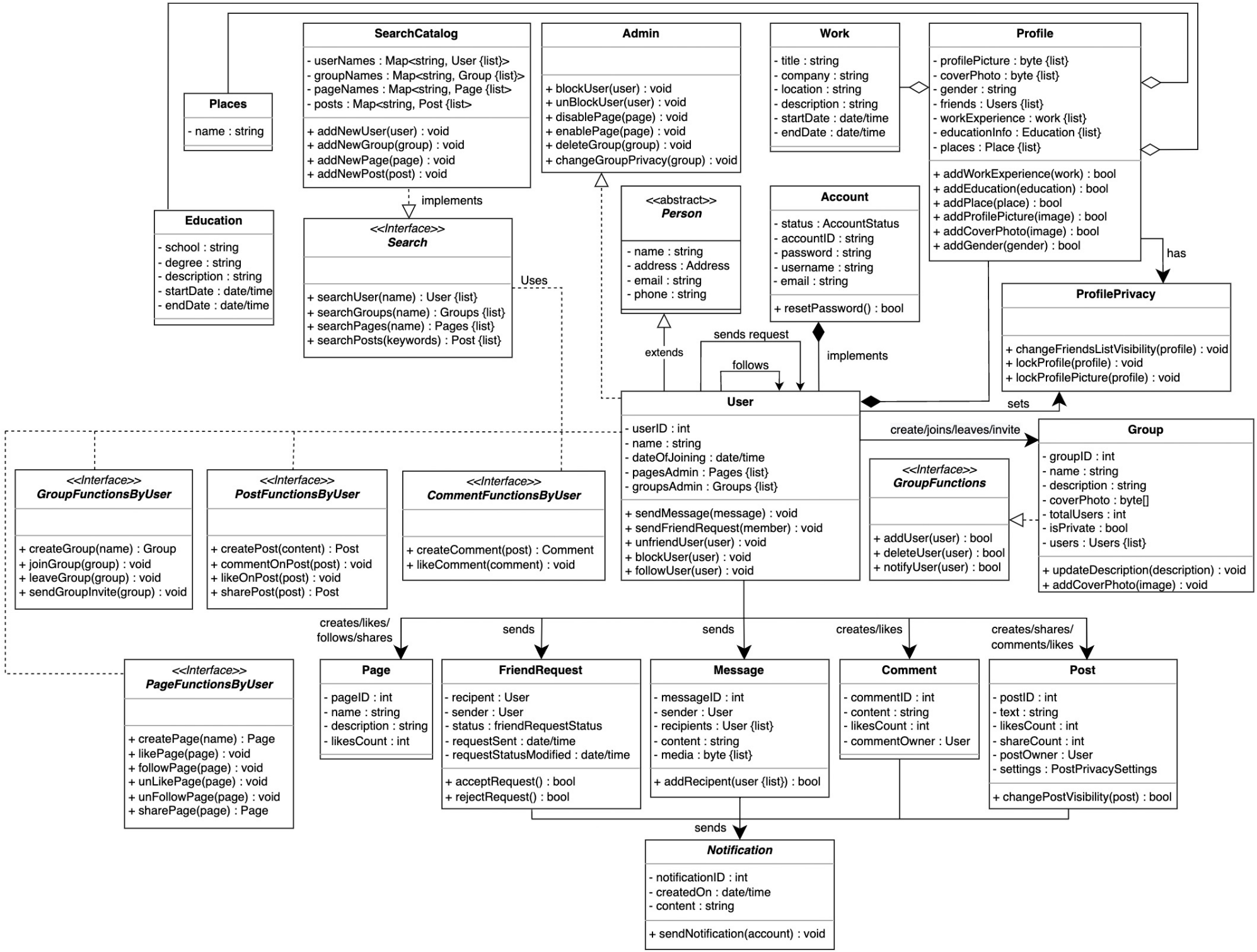
Composition



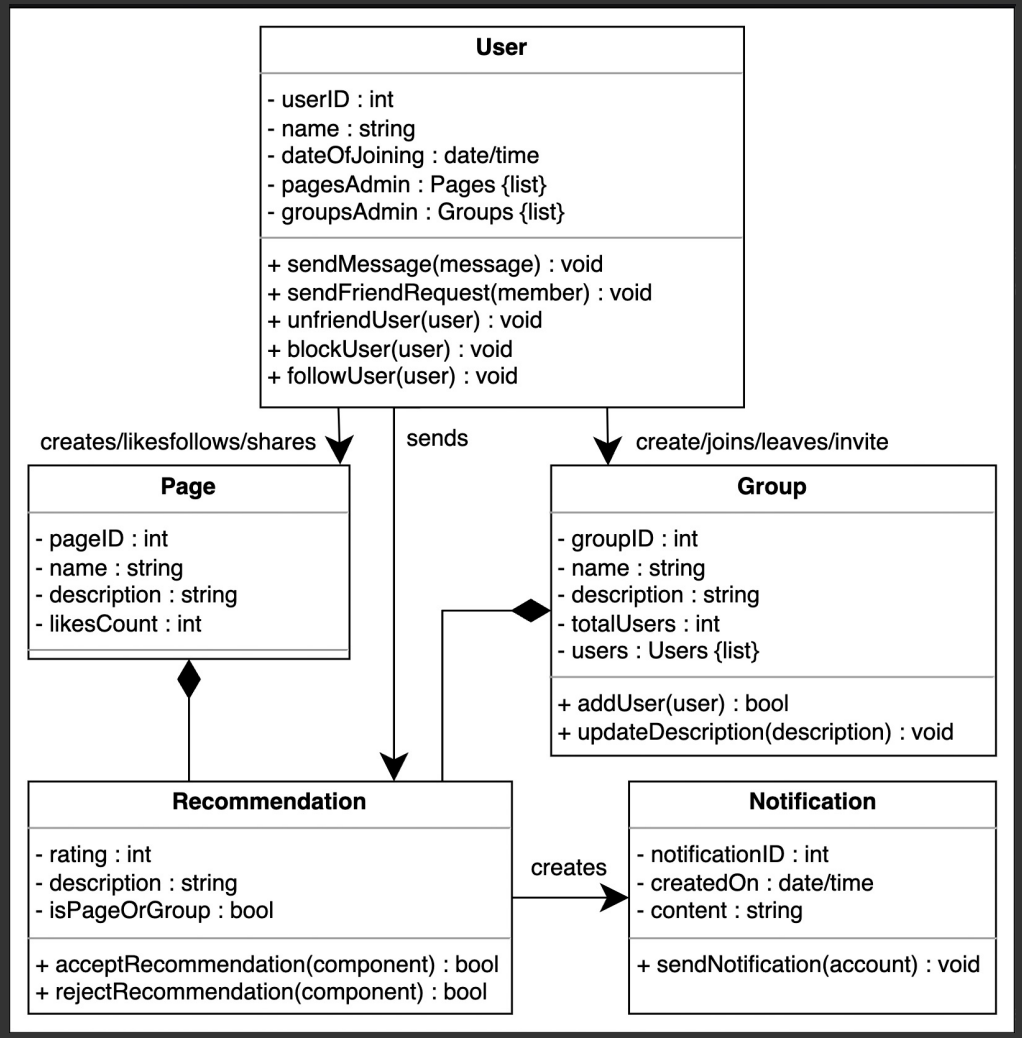
Aggregation

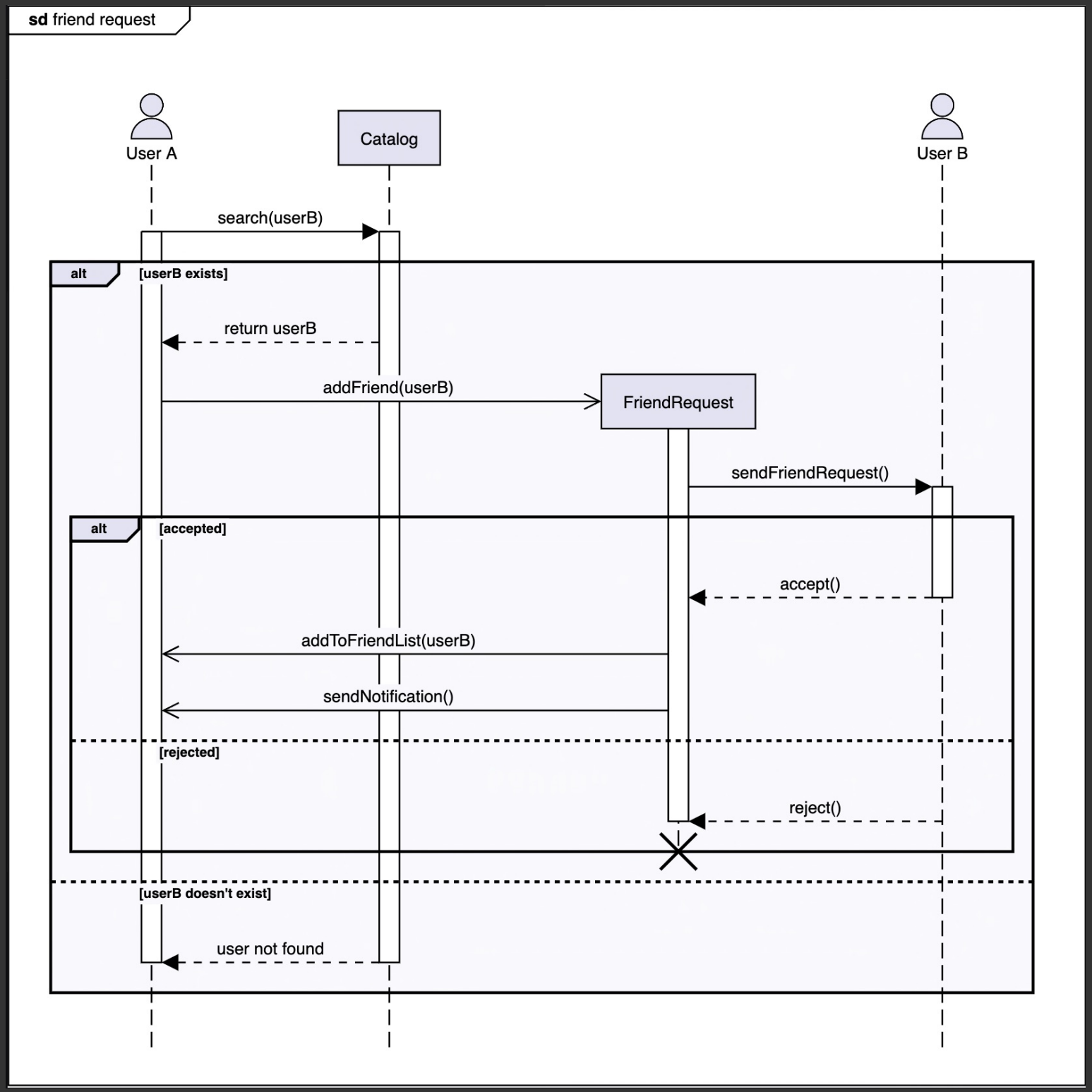


Generalization


SearchCatalog
- userNames : Map<string, User {list}>
- groupNames : Map<string, Group {list}>
- pageNames : Map<string, Page {list>
- posts : Map<string, Post {list>
+ addNewUser(user) : void
+ addNewGroup(group) : void
+ addNewPage(page) : void
+ addNewPost(post) : void
Admin
+ blockUser(user) : void
+ unBlockUser(user) : void
+ disablePage(page) : void
+ enablePage(page) : void
+ deleteGroup(group) : void
+ changeGroupPrivacy(group) : void
Work
- title : string
- company : string
- location : string
- description : string
- startDate : date/time
- endDate : date/time
Profile
- profilePicture : byte {list}
- coverPhoto : byte {list}
- gender : string
- friends : Users {list}
- workExperience : work {list}
- educationInfo : Education {list}
- places : Place {list}
+ addWorkExperience(work) : bool
+ addEducation(education) : bool
+ addPlace(place) : bool
+ addProfilePicture(image) : bool
+ addCoverPhoto(image) : bool
+ addGender(gender) : bool
Places
- name : string
implements
Education
- school : string
- degree : string
- description : string
- startDate : date/time
- endDate : date/time
<<Interface>>
Search
+ searchUser(name) : User {list}
+ searchGroups(name) : Groups {list}
+ searchPages(name) : Pages {list}
+ searchPosts(keywords) : Post {list}
Uses
<<abstract>>
Person
- name : string
- address : Address
- email : string
- phone : string
Account
- status : AccountStatus
- accountID : string
- password : string
- username : string
- email : string
+ resetPassword() : bool
has
ProfilePrivacy
+ changeFriendsListVisibility(profile) : void
+ lockProfile(profile) : void
+ lockProfilePicture(profile) : void
extends
sends request
follows
implements
sets
User
- userID : int
- name : string
- dateOfJoining : date/time
- pagesAdmin : Pages {list}
- groupsAdmin : Groups {list}
+ sendMessage(message) : void
+ sendFriendRequest(member) : void
+ unfriendUser(user) : void
+ blockUser(user) : void
+ followUser(user) : void
create/joins/leaves/invite
Group
- groupID : int
- name : string
- description : string
- coverPhoto : byte[]
- totalUsers : int
- isPrivate : bool
- users : Users {list}
+ updateDescription(description) : void
+ addCoverPhoto(image) : void
<<Interface>>
GroupFunctions
+ addUser(user) : bool
+ deleteUser(user) : bool
+ notifyUser(user) : bool
<<Interface>>
GroupFunctionsByUser
+ createGroup(name) : Group
+ joinGroup(group) : void
+ leaveGroup(group) : void
+ sendGroupInvite(group) : void
<<Interface>>
PostFunctionsByUser
+ createPost(content) : Post
+ commentOnPost(post) : void
+ likeOnPost(post) : void
+ sharePost(post) : Post
<<Interface>>
CommentFunctionsByUser
+ createComment(post) : Comment
+ likeComment(comment) : void
<<Interface>>
PageFunctionsByUser
+ createPage(name) : Page
+ likePage(page) : void
+ followPage(page) : void
+ unLikePage(page) : void
+ unFollowPage(page) : void
+ sharePage(page) : Page
creates/likes/
follows/shares
sends
sends
creates/likes
creates/shares/
comments/likes
Page
- pageID : int
- name : string
- description : string
- likesCount : int
FriendRequest
- recipent : User
- sender : User
- status : friendRequestStatus
- requestSent : date/time
- requestStatusModified : date/time
+ acceptRequest() : bool
+ rejectRequest() : bool
Message
- messageID : int
- sender : User
- recipients : User {list}
- content : string
- media : byte {list}
+ addRecipent(user {list}) : bool
Comment
- commentID : int
- content : string
- likesCount : int
- commentOwner : User
Post
- postID : int
- text : string
- likesCount : int
- shareCount : int
- postOwner : User
- settings : PostPrivacySettings
+ changePostVisibility(post) : bool
sends
Notification
- notificationID : int
- createdOn : date/time
- content : string
+ sendNotification(account) : void

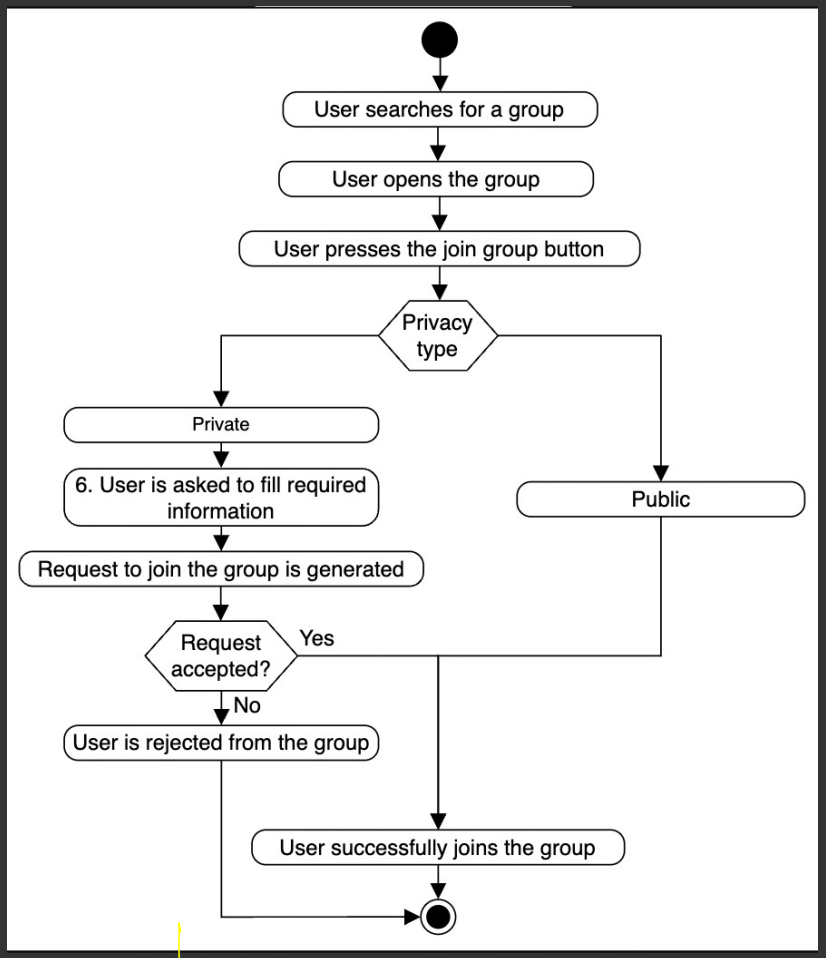
Additional Requirement → Recommendations are added in System



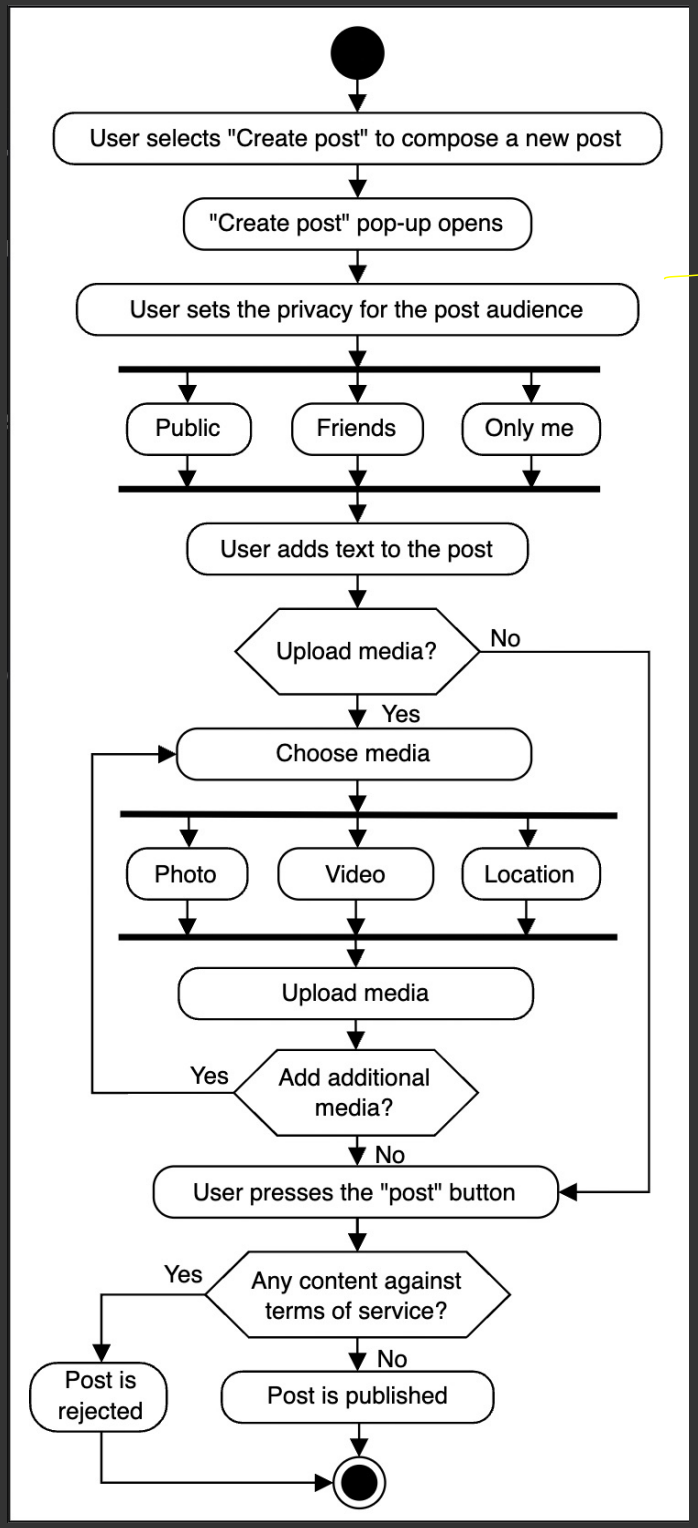


Send a friend Request.

Activity Diagram of FB



when user want to join group



when user want to create a new post.

Code for Facebook

Constant

```
public class Address {
  private int zipCode;
  private String houseNo;
  private String city;
  private String state;
  private String country;
}

enum AccountStatus {
  ACTIVE,
  BLOCKED,
  DISABLED,
  DELETED
}

enum FriendInviteStatus {
  PENDING,
  ACCEPTED,
  REJECTED,
  CANCELED
}

enum PostPrivacySettings {
  PUBLIC,
  FRIENDS_OF_FRIENDS,
  ONLY_FRIENDS,
  CUSTOM
}
```

Interfaces implemented by user

```
public interface PageFunctionsByUser {
  public Page createPage(String name);
  public Page sharePage(Page page);
  public void likePage(Page page);
  public void followPage(Page page);
  public void unLikePage(Page page);
  public void unFollowPage(Page page);
}

public interface GroupFunctionsByUser {
  public Group createGroup(String name);
  public void joinGroup(Group group);
  public void leaveGroup(Group group);
  public void sendGroupInvite(Group group);
}

public interface PostFunctionsByUser {
  public Post createPost(String content);
  public Post sharePost(Post post);
  public void commentOnPost(Post post);
  public void likePost(Post post);
}

public interface CommentFunctionsByUser {
  public Comment createComment(Post post, String
content);
  public void likeComment(Comment comment);
}
```

Account

```
public class Account {
  private String
accountId;
  private String password;
  private String username;
  private String email;
  private AccountStatus
status;

  public boolean
resetPassword();
}
```

Person, User & Admin

```
public class Admin {
  public void blockUser(User user);
  public void unblockUser(User user);
  public void enablePage(Page page);
  public void disablePage(Page page);
  public void deleteGroup(Group group);
  public void changeGroupPrivacy(Group group);
}

// Person will be an abstract class
public abstract class Person {
  private String name;
  private Address address;
  private String email;
  private String phone;
  private Account account;
}
```

```
// Will be using only one interface example
public class User extends Person implements PageFunctionsByUser{
  private int userId;
  private String name;
  private Date dateOfJoining;
  // The following lists contain the pages and groups that a user is
admin of
  private List<Pages> pagesAdmin;
  private List<Groups> groupsAdmin;

  private Profile profile;
  public boolean sendMessage(Message message);
  public boolean sendRecommendation(Page page, Group group, User user);
  public boolean sendFriendRequest(User user);
  public boolean unfriendUser(User user);
  public boolean blockUser(User user);
  public boolean followUser(User user);

  // The functions of the different interfaces will also be present
here
  public Page createPage(String name) {
    // functionality
  }
  public void likePage(Page page){
    // functionality
  }
  public void followPage(Page page){
    // functionality
  }
  public void unLikePage(Page page){
    // functionality
  }
  public void unFollowPage(Page page){
    // functionality
  }
  public Page sharePage(Page page){
    // functionality
  }
}
```

Profile, education, places & work

Places, education, places & work

```
public class Profile {
  private String gender;
  private byte[] profilePicture;
  private byte[] coverPhoto;
  private List<User> friends;

  private List<int> usersFollowed;
  private List<int> pagesFollowed;
  private List<int> groupsJoined;

  private List<Work> workExperience;
  private List<Education> educationInfo;
  private List<Place> places;

  public boolean addWorkExperience(Work work);
  public boolean addEducation(Education education);
  public boolean addPlace(Place place);
  public boolean addProfilePicture(byte[] image);
  public boolean addCoverPhoto(byte[] image);
  public boolean addGender(String gender);
}

public class Work {
  private String title;
  private String company;
  private String location;
  private String description;
  private Date startDate;
  private Date endDate;
}

public class Places {
  private String name;
}

public class Education {
  private String school;
  private String degree;
  private String description;
  private Date startDate;
  private Date endDate;
}
```

Page, post & comment

```
public class Page {
  private int pageId;
  private String name;
  private String description;
  private int likeCount;
}

public class Post {
  private int postId;
  private String content;
  private byte[] image;
  private int likeCount;
  private int shareCount;
  private User postOwner;
  private PostPrivacySettings settings;
  public changePostVisibility(Post post);
}

public class Comment {
  private int commentId;
  private String content;
  private int likeCount;
  private User commentOwner;
}
```

Profile Privacy

```
public class ProfilePrivacy {
    public void changeFriendsListVisibility(Profile
profile) {}
    public void lockProfile(Profile profile) {}
    public void lockProfilePicture(Profile profile)
{}
}
```

Group & group functions interface

```
public interface GroupFunctions {
  public boolean addUser(User user);
  public boolean deleteUser(User user);
  public boolean notifyUser(User user);
}

public class Group implements GroupFunctions {
  private int groupId;
  private String name;
  private String description;
  private byte[] coverPhoto;
  private int totalUsers;
  private boolean isPrivate;
  private List<User> users;

  public boolean addUser(User user) {
    // functionality
  }
  public boolean deleteUser(User user) {
    // functionality
  }
  public boolean notifyUser(User user) {
    // functionality
  }
  public void updateDescription(String description) {}
  public void addCoverPhoto(byte[] image) {}
}
```

Message & friend Request

```
public class Message {
  private int messageId;
  private User sender;
  private String content;
  private List<User> recipent;
  private List<byte[]> multimedia;

  public boolean addRecipent(List<User>
users);
}

public class FriendRequest {
  private User recipent;
  private User sender;
  private FriendRequestStatus status;
  private Date requestSent;
  private Date requestStatusModified;

  public boolean acceptRequest(User user);
  public boolean rejectRequest(User user);
}
```

Notification

```
public class Notification {
  private int notificationId;
  private Date createdOn;
  private String content;

  public boolean sendNotification(Account account);
}
```

Search Catalog & interface

```
public interface Search {
  public List<User> searchUsers(String name);
  public List<Group> searchGroups(String name);
  public List<Page> searchPages(String name);
  public List<Post> searchPosts(String keywords);
}

public class SearchCatalog implements Search {
  HashMap<String, List<User>> userNames;
  HashMap<String, List<Group>> groupNames;
  HashMap<String, List<Page>> pageTitles;
  HashMap<String, List<Post>> posts;

  public boolean addNewUser(User user) {}
  public boolean addNewGroup(Group group) {}
  public boolean addNewPage(Page page) {}
  public boolean addNewPost(Post post) {}

  public List<User> searchUsers(String name) {
    // functionality
  }

  public List<Group> searchGroups(String name) {
    // functionality
  }

  public List<Page> searchPages(String name) {
    // functionality
  }

  public List<Post> searchPosts(String keywords) {
    // functionality
  }
}
```