Table of Contents

Numbers	Content
1	Repo Abstraction
2	Service Abstraction
3	Expected Views (Important)

Repository Abstraction

```
public interface IFriendsRepository
{
    bool FriendshipExists(string userId1, string userId2);
    void AddFriendRequest(string requesterId, string addresseeId);
    void AcceptFriendRequest(string requesterId, string addresseeId);
    void DeclineFriendRequest(string requesterId, string addresseeId);
    void CancelFriendRequest(string requesterId, string addresseeId);
    void BlockUser(string blockerId, string blockedId);
    void Unfriend(string userId1, string userId2);
    int GetFriendCount(string userId);
    int GetPendingRequestCount(string userId);
    IEnumerable<Friends> GetPendingRequests(string userId);
    IEnumerable<Friends> GetFriends(string userId);
    IEnumerable<Friends> GetBlockedUsers(string userId);
    //non friend users
    IEnumerable<User> GetAllUsers();
    FriendStatus GetFriendshipStatus(string userId1, string userId2);
    IEnumerable<User> GetPeopleYouMayKnow(string currentUserId);
    public int GetMutualFriendCount(string currentUserId, string otherUserId);
}
```

→ P Y M[illegible]

People you may know

```
}

public interface IPostRepository
{
   Task<Post> GetByIdAsync(int postId);
   Task<Post> CreateAsync(Post post);
   Task UpdateAsync(Post post);
   Task DeleteAsync(int postId);
   Task<IEnumerable<Post>> GetByUserIdAsync(string userId);
   Task<IEnumerable<Post>> GetRecentPostsAsync(int count = 10);
   Task<IEnumerable<Post>> GetPopularPostsAsync(TimeSpan since);
   Task<bool> ExistsAsync(int postId);
   Task<bool> IsOwnerAsync(int postId, string userId);
   Task<int> GetPostCountAsync(string userId = null);
   Task<IEnumerable<Post>> GetDeletedPostsAsync();
}

public interface IPostImagesRepository
 {
    Task<PostImages> GetImageByImageIdAsync(int imageId);
    Task<IEnumerable<PostImages>> GetImageByPostIdAsync(int postId);
    Task<PostImages> AddPostImageAsync(PostImages image);
    Task UpdateImagePathAsync(int imageId, string newPath);
    Task DeleteImageAsync(int imageId);
    Task AddRangeAsync(IEnumerable<PostImages> images);
    Task<IEnumerable<PostImages>> GetDeletedImagesAsync(DateTime? since =
null);
    Task<int> GetImageCountForPostAsync(int postId);
 }
```

```
public interface IPostReactionsRepository
{
    Task<PostReactions> GetReactionAsync(int postId, string userId);
    Task AddReactionAsync(PostReactions reaction);
    Task UpdateReactionAsync(int reactionId, ReactionTypes newReactionType);
    Task RemoveReactionAsync(int reactionId);
    Task<bool> HasUserReactedAsync(int postId, string userId);
    Task<int> GetReactionCountAsync(int postId, ReactionTypes? type = null);
    Task<IEnumerable<PostReactions>> GetReactionsByPostAsync(int postId,
bool includeDeleted = false);
    Task<IEnumerable<PostReactions>> GetReactionsByUserAsync(string userId,
bool includeDeleted = false);
}
```

```
public interface IPostReactionsRepository
{
    Task<PostReactions> GetReactionAsync(int postId, string userId);
    Task AddReactionAsync(PostReactions reaction);
    Task UpdateReactionAsync(int reactionId, ReactionTypes newReactionType);
    Task RemoveReactionAsync(int reactionId);
    Task<bool> HasUserReactedAsync(int postId, string userId);
    Task<int> GetReactionCountAsync(int postId, ReactionTypes? type = null);
    Task<IEnumerable<PostReactions>> GetReactionsByPostAsync(int postId,
bool includeDeleted = false);
    Task<IEnumerable<PostReactions>> GetReactionsByUserAsync(string userId,
bool includeDeleted = false);}
```

```
public interface IUserRepository
```

```
{
    Task<bool> RegisterUserAsync(User user, string password);
    Task<bool> DeleteUserAsync(User user);
    Task<bool> ChangeUserPasswordAsync(User user, string oldPassword, string
newPassword);
    Task<IdentityResult> UpdateUserAsync(User user);
    Task<User> FindByEmailAsync(string email);
    Task<bool> CheckPasswordAsync(User user, string password);
}
```

Services Abstraction:

public interface IAuthenticationService

```
{
    bool IsSignedIn(ClaimsPrincipal user);
    Task<User> LoginAsync(string email, string password, bool rememberMe);
    Task LogoutAsync();
    Task<bool> IsEmailConfirmedAsync(string email);
}


public interface IFriendsService
{
    bool FriendshipExists(string userId1, string userId2);
    void AddFriendRequest(string requesterId, string addresseeId);
    void AcceptFriendRequest(string requesterId, string addresseeId);
    void DeclineFriendRequest(string requesterId, string addresseeId);
    void CancelFriendRequest(string requesterId, string addresseeId);
    void BlockUser(string blockerId, string blockedId);
    void Unfriend(string userId1, string userId2);
    int GetFriendCount(string userId);
    int GetPendingRequestCount(string userId);
    IEnumerable<Friends> GetPendingRequests(string userId);
    IEnumerable<Friends> GetFriends(string userId);
    IEnumerable<Friends> GetBlockedUsers(string userId);
    FriendStatus GetFriendshipStatus(string userId1, string userId2);
    IEnumerable<PoepleMV> GetAllUsers();
    IEnumerable<PoepleMV> GetPeopleYouMayKnow(string userId);
    IEnumerable<PoepleMV> MyConnections(string userId);
}
```

```
public interface IPostCommentsService
 {
    Task<PostComments> GetCommentAsync(int commentId);
    Task<PostComments> CreateCommentAsync(int postId, string commenterId,
string content, string? imgPath = null, int? parentCommentId = null);
    Task UpdateCommentAsync(int commentId, string newContent, string?
newImgPath = null);
    Task DeleteCommentAsync(int commentId);
    Task<PostComments> ReplyToCommentAsync(int parentCommentId, string
commenterId, string content, string? imgPath = null);
    Task<IEnumerable<PostComments>> GetCommentsForPostAsync(int postId);
    Task<IEnumerable<PostComments>> GetCommentRepliesAsync(int
parentCommentId);
    Task<bool> IsCommentOwnerAsync(int commentId, string userId);
    Task<int> GetCommentCountForPostAsync(int postId);
 }
```

```
public interface IPostImagesService
{
   Task<PostImages> GetImageByImageIdAsync(int imageId);
   Task<IEnumerable<PostImages>> GetImageByPostIdAsync(int postId);
   Task<PostImages> AddPostImageAsync(PostImages image);
   Task UpdateImagePathAsync(int imageId, string newPath);
   Task DeleteImageAsync(int imageId);
   Task AddRangeAsync(IEnumerable<PostImages> images);
   Task<IEnumerable<PostImages>> GetDeletedImagesAsync(DateTime? since =
null);
```

```
    Task<int> GetImageCountForPostAsync(int postId);
}


public interface IPostReactionsService
 {
     Task ToggleReactionAsync(int postId, string userId, ReactionTypes
reactionType);
 }


public interface IPostService
{
    Task<Post> GetPostByIdAsync(int postId);
    Task<Post> CreatePostAsync(string userId, string textContent);
    Task UpdatePostAsync(int postId, string textContent);
    Task DeletePostAsync(int postId);
    Task<IEnumerable<Post>> GetUserPostsAsync(string userId);
    Task<int> GetUserPostCountAsync(string userId);
    Task<IEnumerable<Post>> GetRecentPostsAsync(int count = 10);
    Task<IEnumerable<Post>> GetPopularPostsAsync(TimeSpan since);
    Task<bool> IsPostOwnerAsync(int postId, string userId);
}

public interface IUserService
{
    Task<bool> RegisterUserAsync(UserRegisterMV user, string password);
    Task<bool> DeleteUserAsync(UserMV user);
    Task<bool> UpdateUserAsync(UserMV oldUser, UserMV newUser);
    Task<bool> ChangeUserPasswordAsync(UserMV user, string oldPassword,
string newPassword);


}
```

Expectations/Suggestions/(that’s the beginning)

2. User Profile Views

A. Profile View

Methods: IUserRepository.FindByEmailAsync, IFriendsService.GetFriendCount
Elements:

- Profile header with:
    - Profile picture
    - User name
    - Bio/description
    - Friend count (from GetFriendCount)
- Tab navigation for:
    - Posts
    - Friends
    - Photos
- [Edit Profile] button (for owner)

B. Edit Profile View

Methods: IUserService.UpdateUserAsync, IUserService.ChangeUserPasswordAsync
Elements:

- Form with editable fields:
    - Profile picture upload
    - Name
    - Bio
    - Email
- Change password section:
    - Current password
    - New password
    - Confirm new password

- [Save Changes] button
- [Cancel] button

3. Social/Connection Views

A. Friends List View

Methods: IFriendsService.GetFriends, IFriendsService.GetPendingRequests
Elements:

- Two tabs:
    1. Current Friends (list from GetFriends)
    2. Pending Requests (list from GetPendingRequests)
- Each friend item shows:
    - Profile picture
    - Name
    - Three dots list contains options like : unfriend
- Pending requests show:
    - Requester info
    - [Accept]/[Decline] buttons

- [Login] button
- "Forgot password" link
- Link to Registration page

B. People Search/Discovery View

Methods: IFriendsService.GetAllUsers, IFriendsService.GetPeopleYouMayKnow
Elements:

- Search bar
- "People You May Know" section (from GetPeopleYouMayKnow)
- "All Users" section
- Each user card shows:
    - Profile picture
    - Name
    - Mutual friends count
    - [Add Friend] buttons

4. Post/Content Views

A. News Feed View

Methods: IPostService.GetRecentPostsAsync, IPostService.GetPopularPostsAsync
Elements:

- Create Post component (textarea + Add Image Button + [Post] button)
- Tab navigation:
    - Recent Posts (from GetRecentPostsAsync)
    - Popular Posts (from GetPopularPostsAsync)
    - Saved Posts
- Each post shows:

    - Author info
    - Post content
    - Images (if any)
    - Like/comment counts
    - Reaction button
    - Timestamp
    - Comment Button
    - Share Button
    - Three dotes list contain options like: Save Post, Delete Post(for owner), Edit

B. Single Post View

Methods: IPostService.GetPostByIdAsync, IPostCommentsService.GetCommentsForPostAsync
Elements:

- Full post display
- Comment section:
    - Comment input box
    - List of comments (from GetCommentsForPostAsync)
    - Reply functionality (nested comments)
- Reaction buttons

C. Create/Edit Post View

Methods: IPostService.CreatePostAsync, IPostService.UpdatePostAsync
Elements:

- Textarea for post content
- Image upload button
- Privacy selector
- [Post]/[Update] button

5. Comment Views

A. Comment Component

Methods: IPostCommentsService.CreateCommentAsync, IPostCommentsService.GetCommentRepliesAsync
Elements:

- Comment textarea
- [Post Comment] button
- Nested replies display
- [Reply] button on each comment
- [Edit]/[Delete] buttons (for comment owner)

B. Content Moderation View

Methods: IPostRepository.GetDeletedPostsAsync, IPostImagesRepository.GetDeletedImagesAsync
Elements:

- Tabs for:
  - Deleted posts
  - Deleted images
  - Reported content
- Restoration options

- Permanent deletion options