Back To Course Home

Grokking the Object Oriented Design Interview

0% completed

Search Course

Object-Oriented Design and UML

- Object-Oriented Basics
- OO Analysis and Design
- What is UML?
- Use Case Diagrams
- Class Diagram
- Sequence diagram
- Activity Diagrams

Object Oriented Design Case Studies

- Design a Library Management System
- Design a Parking Lot
- Design Amazon - Online Shopping System
- Design Stack Overflow
- Design a Movie Ticket Booking System
- Design an ATM
- Design an Airline Management System
- Design Blackjack and a Deck of Cards
- Design a Hotel Management System
- Design a Restaurant Management system
- Design Chess
- Design an Online Stock Brokerage System
- Design a Car Rental System
- Design LinkedIn
- Design Cricinfo
- Design Facebook - a social network

Appendix

- Contact us
- Other courses

Course Certificate

Mark Course as Completed

Design Facebook - a social network

We'll cover the following

- System Requirements
- Use case diagram
- Class diagram
- Activity diagrams
- Code
- Extended requirement

Facebook is an online social networking service where users can connect with other users to post and read messages. Users access Facebook through their website interface or mobile apps.

System Requirements

We will focus on the following set of requirements while designing Facebook:

1. Each member should be able to add information about their basic profile, work experience, education, etc.
2. Any user of our system should be able to search other members, groups or pages by their name.
3. Members should be able to send and accept/reject friend requests from other members.
4. Members should be able to follow other members without becoming their friend.
5. Members should be able to create groups and pages, as well as join already created groups, and follow pages.
6. Members should be able to create new posts to share with their friends.
7. Members should be able to add comments to posts, as well as like or share a post or comment.
8. Members should be able to create privacy lists containing their friends. Members can link any post with a privacy list to make the post visible only to the members of that list.
9. Any member should be able to send messages to other members.
10. Any member should be able to add a recommendation for any page.
11. The system should send a notification to a member whenever there is a new message or friend request or comment on their post.
12. Members should be able to search through posts for a word.

Extended Requirement: Write a function to find a connection suggestion for a member.

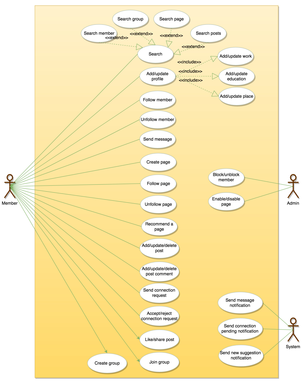
Use case diagram

We have three main Actors in our system:

- **Member:** All members can search for other members, groups, pages, or posts, as well as send friend requests, create posts, etc.
- **Admin:** Mainly responsible for admin functions like blocking and unblocking a member, etc.
- **System:** Mainly responsible for sending notifications for new messages, friend requests, etc.

Here are the top use cases of our system:

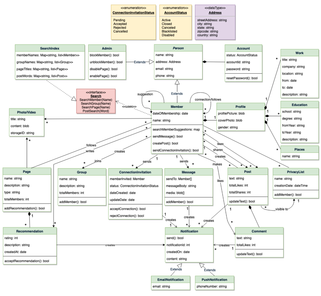
- **Add/update profile:** Any member should be able to create their profile to reflect their work experiences, education, etc.
- **Search:** Members can search for other members, groups or pages. Members can send a friend request to other members.
- **Follow or Unfollow a member or a page:** Any member can follow or unfollow any other member or page.
- **Send message** Any member can send a message to any of their friends.
- **Create post** Any member can create a post to share with their friends, as well as like or add comments to any post visible to them.
- **Send notification** The system will be able to send notifications for new messages, friend requests, etc.

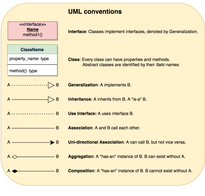


Class diagram

Here are the main classes of the Facebook system:

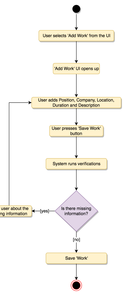
- **Member:** This will be the main component of our system. Each member will have a profile which includes their Work Experiences, Education, etc. Members will be connected to other members and they can follow other members and pages. Members will also have suggestions to send friend requests to other members.
- **Search:** Our system will support searching for other members, groups and pages by their names, and through posts for any word.
- **Message:** Members can send messages to other members with text, photos, and videos.
- **Post:** Members can create posts containing text and media, as well as like and share a post.
- **Comment:** Members can add comments to posts as well as like any comment.
- **Group:** Members can create and join groups.
- **PrivacyList:** Members can create privacy lists containing their friends. Members can link any post with a privacy list, to make the post visible only to the members of that list.
- **Page:** Members can create pages that other members can follow, and share messages there.
- **Notification:** This class will take care of sending notifications to members. The system will be able to send a push notification or an email.



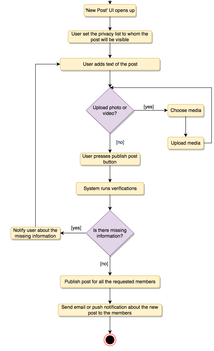


Activity diagrams

Add work experience to profile: Any Facebook member can perform this activity. Here are the steps to add work experience to a member's profile:



Create a new post: Any Member can perform this activity. Here are the steps for creating a post:



Code

Here is the high-level definition for the classes described above.

Enums, data types, and constants: Here are the required enums, data types, and constants:

Java | Python

```
public enum ConnectionInvitationStatus{
  PENDING,
  ACCEPTED,
  REJECTED,
  CANCELED
}

public enum AccountStatus{
  ACTIVE,
  CLOSED,
  CANCELED,
  BLACKLISTED,
  DISABLED
}

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

Account, Person, Member, and Admin: These classes represent the different people that interact with our system:

Java | Python

```
// For simplicity, we are not defining getter and setter functions. The reader can
// assume that all class attributes are private and accessed through their respective
// public getter method and modified only through their public setter method.

public class Account {
  private String id;
  private String password;
  private AccountStatus status;

  public boolean resetPassword();
}

public abstract class Person {
  private String name;
  private Address address;
  private String email;
  private String phone;

  private Account account;
}

public class Member extends Person {
  private Integer memberId;
  private Date dateOfMembership;
  private String name;

  private Profile profile;
  private HashSet<Integer> memberFollows;
```

Profile and Work: A member's profile will have their work experiences, educations, places, etc:

Java | Python

```
public class Profile {
  private byte[] profilePicture;
  private byte[] coverPhoto;
  private String gender;

  private List<Work> workExperiences;
  private List<Education> educations;
  private List<Place> places;
  private List<Stat> stats;

  public boolean addWorkExperience(Work work);
  public boolean addEducation(Education education);
  public boolean addPlace(Place place);
}

public class Work {
  private String title;
  private String company;
  private String location;
  private Date from;
  private Date to;
  private String description;
}
```

Page and Recommendation: Each page can have multiple recommendations, and members will follow/like pages:

Java | Python

```
public class Page {
  private Integer pageId;
  private String name;
  private String description;
  private String type;
  private int totalMembers;
  private List<Recommendation> recommendation;

  private List<Recommendation> getRecommendation();
}

public class Recommendation {
  private Integer recommendationId;
  private int rating;
  private String description;
  private Date createdAt;
}
```

Group, Post, Message, and Comment: Members can create posts, comment on posts, send messages and join groups:

Java | Python

```
public class Group {
  private Integer groupId;
  private String name;
  private String description;
  private int totalMembers;
  private List<Member> members;

  public boolean addMember(Member member);
  public boolean updateDescription(String description);
}

public class Post {
  private Integer postId;
  private String text;
  private int totalLikes;
  private int totalShares;
  private Member owner;
}

public class Message {
  private Integer messageId;
  private Member[] sentTo;
  private String messageBody;
  private byte[] media;

  public boolean addMember(Member member);
}
```

Search interface and SearchIndex: SearchIndex will implement Search to facilitate searching of members, groups, pages, and posts:

Java | Python

```
public interface Search {
  public List<Member> searchMember(String name);
  public List<Group> searchGroup(String name);
  public List<Page> searchPage(String name);
  public List<Post> searchPost(String word);
}

public class SearchIndex implements Search {
  HashMap<String, List<Member>> memberNames;
  HashMap<String, List<Group>> groupNames;
  HashMap<String, List<Page>> pageTitles;
  HashMap<String, List<Post>> posts;

  public boolean addMember(Member member) {
    if(memberNames.containsKey(member.getName())) {
      memberNames.get(member.getName()).add(member);
    } else {
      memberNames.put(member.getName(), member);
    }
  }

  public boolean addGroup(Group group);
  public boolean addPage(Page page);
  public boolean addPost(Post post);

  public List<Member> searchMember(String name) {
    return memberNames.get(name);
  }
```

Extended requirement

Here is the code for finding connection suggestions for a member.

There can be many strategies to search for connection suggestions; we will do a two-level deep breadth-first search to find people who have the most connections with each other. These people could be good candidates for a connection suggestion, here is the sample Java code:

```
import java.util.HashSet;
import java.util.LinkedHashMap;
import java.util.Map;
import java.util.stream.Collectors;
import static java.util.Collections.reverseOrder;

public class Member extends Person {
  private Integer memberId;
  private Date dateOfMembership;
  private String name;

  private Profile profile;
  private HashSet<Integer> memberFollows;
  private HashSet<Integer> memberConnections;
  private HashSet<Integer> pageFollows;
  private HashSet<Integer> memberSuggestions;
  private HashSet<ConnectionInvitation> connectionInvitations;
  private HashSet<Integer> groupFollows;

  public boolean sendMessage(Message message);
  public boolean createPost(Post post);
  public boolean sendConnectionInvitation(ConnectionInvitation invitation);

  private Map<Integer, Integer> searchMemberSuggestions() {
    Map<Integer, Integer> suggestions = new HashMap<>();
    for(Integer memberId : this.memberConnections) {
      HashSet<Integer> firstLevelConnections = new Member(memberId).getMemberConnections();
      for(Integer firstLevelConnectionId : firstLevelConnections) {
```