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 Stack Overflow

We'll cover the following

- Requirements and Goals of the System
- Use-case Diagram
- Class diagram
- Activity diagrams
- Sequence Diagram
- Code

Stack Overflow is one of the largest online communities for developers to learn and share their knowledge. The website provides a platform for its users to ask and answer questions, and through membership and active participation, the users vote on questions and answers up or down. Users can edit questions and answers in a fashion similar to a wiki.

Users of Stack Overflow can earn reputation points and badges. For example, a person is awarded ten reputation points for receiving an "up" vote on an answer and five points for the "up" vote of a question. The can also receive badges for their valued contributions. A higher reputation lets users unlock new privileges like the ability to vote, comment on, and even edit other people's posts.

Requirements and Goals of the System

We will be designing a system with the following requirements:

1. Any non-member (guest) can search and view questions. However, to add or upvote a question, they have to become a member.
2. Members should be able to post new questions.
3. Members should be able to add an answer to an open question.
4. Members can add comments to any question or answer.
5. A member can upvote a question, answer or comment.
6. Members can flag a question, answer or comment, for serious problems or moderator attention.
7. Any member can add a bounty to their question to draw attention.
8. Members will earn badges for being helpful.
9. Members can vote to close a question; Moderators can close or reopen any question.
10. Members can add tags to their questions. A tag is a word or phrase that describes the topic of the question.
11. Members can vote to delete extremely off-topic or very low-quality questions.
12. Moderators can close a question or undelete an already deleted question.
13. The system should also be able to identify most frequently used tags in the questions.

Use-case Diagram

We have five main actors in our system:

- **Admin:** Mainly responsible for blocking or unblocking members.
- **Guest:** All guests can search and view questions.
- **Member**: Members can perform all activities that guests can, in addition to which they can add/remove questions, answers, and comments. Members can delete and un-delete their questions, answers or comments.
- **Moderator:** In addition to all the activities that members can perform, moderators can close/delete/undelete any question.
- **System:** Mainly responsible for sending notifications and assigning badges to members.

Here are the top use cases for Stack Overflow:

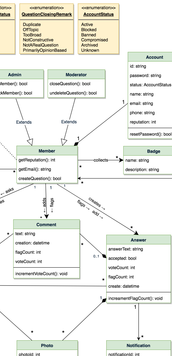
1. Search questions.
2. Create a new question with bounty and tags.
3. Add/modify answers to questions.
4. Add comments to questions or answers.
5. Moderators can close, delete, and un-delete any question.

Use case diagram

Class diagram

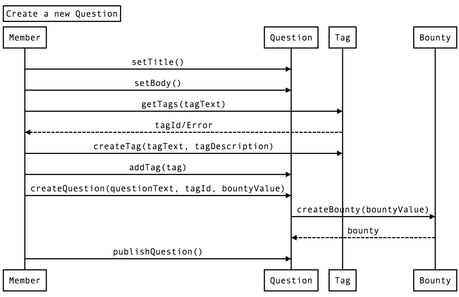
Here are the main classes of Stack Overflow System:

- **Question:** This class is the central part of our system. It has attributes like Title and Description to define the question. In addition to this, we will track the number of times a question has been viewed or voted on. We should also track the status of a question, as well as closing remarks if the question is closed.
- **Answer:** The most important attributes of any answer will be the text and the view count. In addition to that, we will also track the number of times an answer is voted on or flagged. We should also track if the question owner has accepted an answer.
- **Comment:** Similar to answer, comments will have text, and view, vote, and flag counts. Members can add comments to questions and answers.
- **Tag:** Tags will be identified by their names and will have a field for a description to define them. We will also track daily and weekly frequencies at which tags are associated with questions.
- **Badge:** Similar to tags, badges will have a name and description.
- **Photo:** Questions or answers can have photos.
- **Bounty:** Each member, while asking a question, can place a bounty to draw attention. Bounties will have a total reputation and an expiry date.
- **Account:** We will have four types of accounts in the system, guest, member, admin, and moderator. Guests can search and view questions. Members can ask questions and earn reputation by answering questions and from bounties.
- **Notification:** This class will be responsible for sending notifications to members and assigning badges to members based on their reputations.


«enumeration»
QuestionStatus
Open
Closed
Onhold
Deleted
«enumeration»
QuestionClosingRemark
Duplicate
Off-Topic
TooBroad
NotConstructive
AskedAndAnswered
PrimarilyOpinionBased
«enumeration»
AccountStatus
Active
Blocked
Banned
Compromised
Archived
Unknown
Guest
registerAccount(): bool
«abstract»
Search
Admin
blockMember(): bool
unblockMember(): bool
Moderator
closeQuestion(): bool
undeleteQuestion(): bool
Account
id: string
password: string
status: AccountStatus
name: string
email: string
phone: string
reputation: int
resetPassword(): bool
Tag
name: string
description: string
dailyAskedFrequency: int
weeklyAskedFrequency: int
Member
getReputation(): int
getEmail(): string
createQuestion(): bool
Badge
name: string
description: string
Question
title: string
description: string
viewCount: int
voteCount: int
creationTime: datetime
updateTime: datetime
status: QuestionStatus
closingRemark: QuestionClosingRemark
close(): bool
undelete(): bool
Comment
text: string
creation: datetime
flagCount: int
voteCount: int
incrementVoteCount(): void
Answer
answerText: string
accepted: bool
voteCount: int
flagCount: int
create: datetime
incrementFlagCount(): void
Bounty
reputation: int
expiry: dateTime
modifyReputation(): bool
Photo
photoId: int
photoPath: string
creationDate: dateTime
delete(): bool
Notification
notificationId: int
createdOn: date
content: string
sendNotification(): bool
Extends
Extends
creates
collects
has_a_bounties
offers
has_a_photos
has_a_tags
has_a_comments
has_a_photos
has_a_tags
has_a_comments
for
1
0..1
0..1
0..1
0..1
1
1

Sequence Diagram

Following is the sequence diagram for creating a new question:



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

```java
public enum QuestionStatus{
  OPEN,
  CLOSED,
  ON_HOLD,
  DELETED
}

public enum QuestionClosingRemark{
  DUPLICATE,
  OFF_TOPIC,
  TOO_BROAD,
  NOT_CONSTRUCTIVE,
  NOT_A_REAL_QUESTION,
  PRIMARILY_OPINION_BASED
}

public enum AccountStatus{
  ACTIVE,
  CLOSED,
  CANCELED,
  BLACKLISTED,
  BLOCKED
}
```

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

Java | Python

```java
// 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 methods and modified only through their public methods function.

public class Account {
  private String id;
  private String password;
  private AccountStatus status;
  private String name;
  private Address address;
  private String email;
  private String phone;
  private int reputation;

  public boolean resetPassword();
}

public class Member {
  private Account account;
  private List<Badge> badges;

  public int getReputation();
  public String getEmail();
  public boolean createQuestion(Question question);
  public boolean createTag(Tag tag);
}

public class Admin extends Member {
```

Badge, Tag, and Notification: Members have badges, questions have tags and notifications:

Java | Python

```java
public class Badge {
  private String name;
  private String description;
}

public class Tag {
  private String name;
  private String description;
  private long dailyAskedFrequency;
  private long weeklyAskedFrequency;
}

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

  public boolean sendNotification();
}
```

Photo and Bounty: Members can put bounties on questions. Answers and Questions can have multiple photos:

Java | Python

```java
public class Photo {
  private int photoId;
  private String photoPath;
  private Date creationDate;

  private Member creatingMember;

  public boolean delete();
}

public class Bounty {
  private int reputation;
  private Date expiry;

  public boolean modifyReputation(int reputation);
}
```

Question, Comment and Answer: Members can ask questions, as well as add an answer to any question. All members can add comments to all open questions or answers:

Java | Python

```java
public interface Search {
  public static List<Question> search(String query);
}

public class Question implements Search {
  private String title;
  private String description;
  private int viewCount;
  private int voteCount;
  private Date creationTime;
  private Date updateTime;
  private QuestionStatus status;
  private QuestionClosingRemark closingRemark;

  private Member askingMember;
  private Bounty bounty;
  private List<Photo> photos;
  private List<Comment> comments;
  private List<Answer> answers;

  public boolean close();
  public boolean undelete();
  public boolean addComment(Comment comment);
  public boolean addBounty(Bounty bounty);

  public static List<Question> search(String query) {
    // return all questions containing the string query in their title or description.
  }
```