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, to vote questions and answers up or down. Users can edit questions and answers in a fashion similar to a wiki (https://en.wikipedia.org/wiki/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.
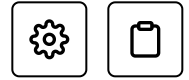
Requirements and Goals of the System

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 (https://stackoverflow.com/help/bounty) to their question to draw attention.

8. Members will earn badges (https://stackoverflow.com/help/badges) for being helpful.

9. Members can vote to close (https://stackoverflow.com/help/closed-questions) a question; Moderators can close or reopen any question.

10. Members can add tags (https://stackoverflow.com/help/tagging) to their questions. A tag is a word or phrase that describes the topic of the question.

11. Members can vote to delete (https://stackoverflow.com/help/deleted-questions) 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.
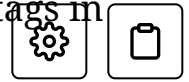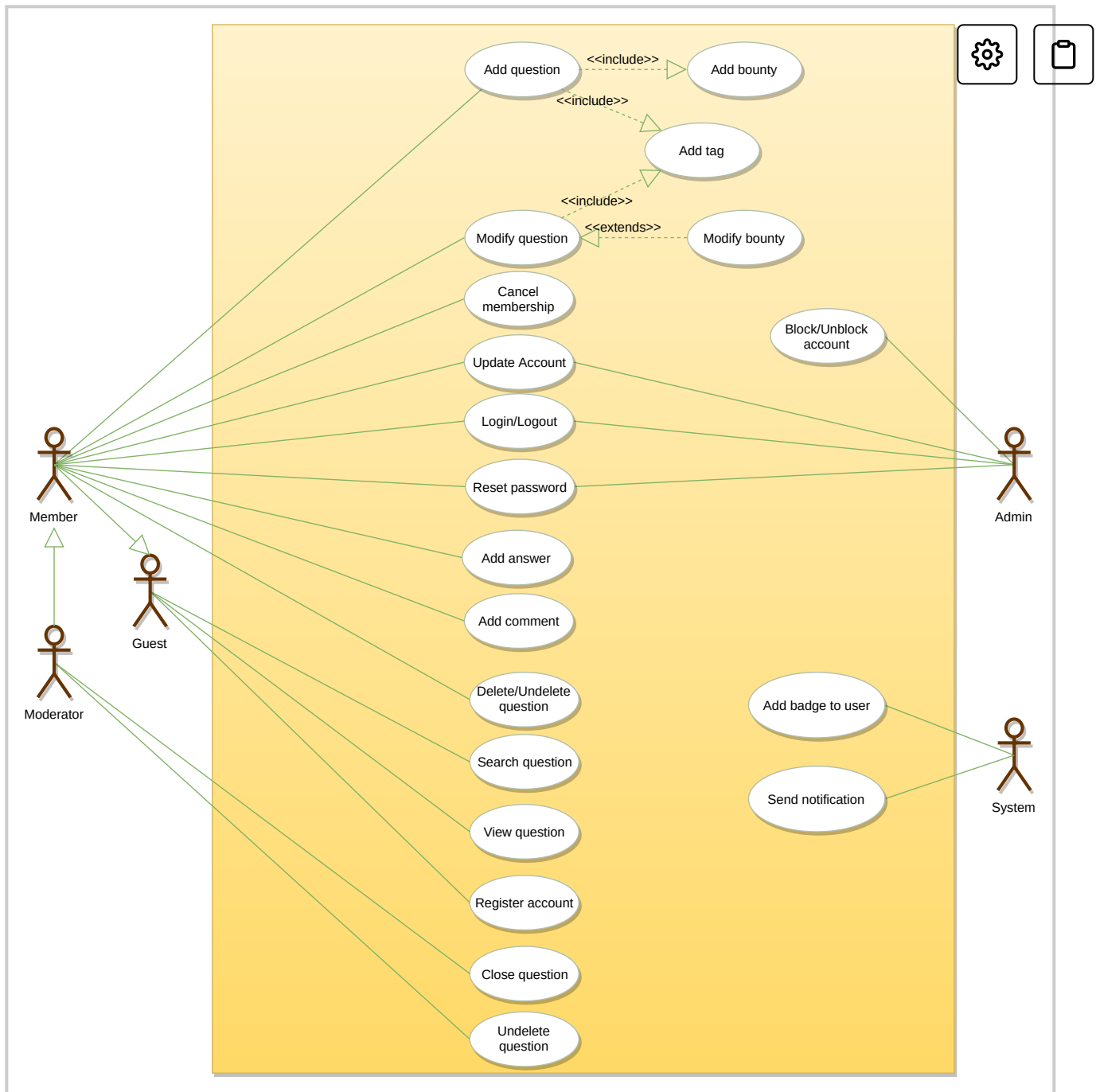
Use-case Diagram

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
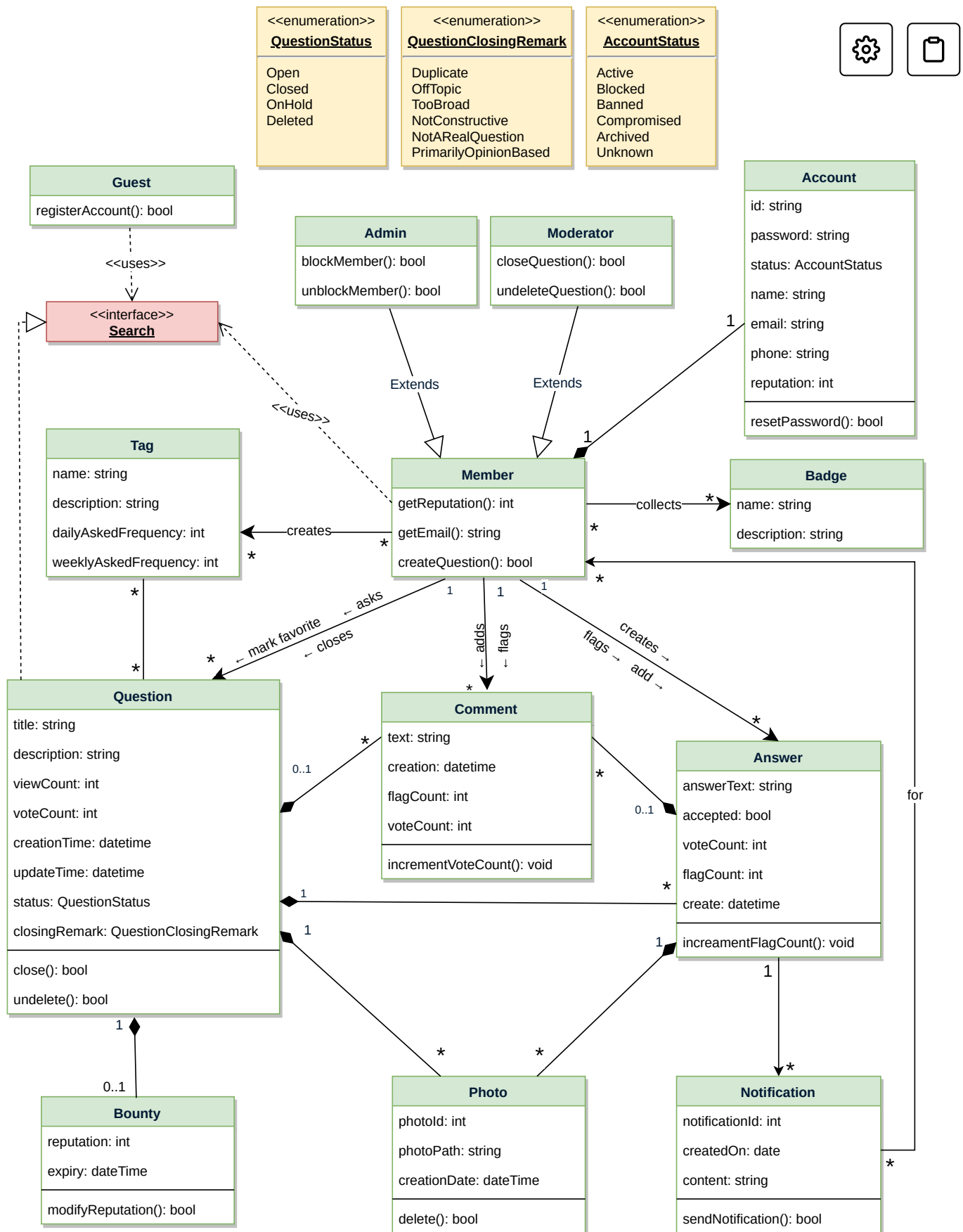
Class diagram
#

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.
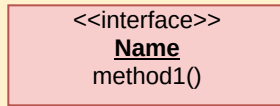
We should also track the status of a question, as well as closing remarks if the question is closed.
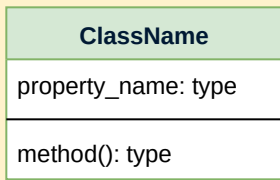
- **Answer:** The most important attributes of any answer will be the text and the view count. In addition to that, we will also track the number of times an answer is voted on or flagged. We should also track if the question owner has accepted an answer.

- **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
OffTopic
TooBroad
NotConstructive
NotARealQuestion
PrimarilyOpinionBased

<<enumeration>>
AccountStatus

Active
Blocked
Banned
Compromised
Archived
Unknown

Guest

registerAccount(): bool

<<uses>>

<<interface>>
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

Extends   Extends

<<uses>>

Tag

name: string

description: string

dailyAskedFrequency: int

weeklyAskedFrequency: int

Member

getReputation(): int

getEmail(): string

createQuestion(): bool

creates

collects

Badge

name: string

description: string

← asks
← mark favorite   ← closes
adds ↓
flags ↓
flags →   creates →
add →

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

increamentFlagCount(): 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

for

Class diagram

UML conventions

<<interface>>
Name
method1()

Interface: Classes implement interfaces, denoted by Generalization.

ClassName
property_name: type
method(): type

Class: Every class can have properties and methods.
Abstract classes are identified by their *Italic* names.
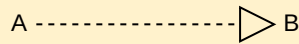
A - - - - - - - - - - - ▷ B   **Generalization**: A implements B.
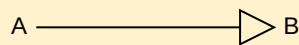
A ————————▷ B   **Inheritance**: A inherits from B. A "is-a" B.
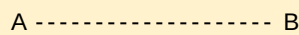
A - - - - - - - - - - - B   **Use Interface:** A uses interface B.
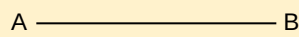
A ———————— B   **Association**: A and B call each other.
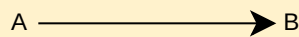
A ————————▶ B   **Uni-directional Association**: A can call B, but not vice versa.
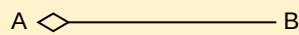
A ◇———————— B   **Aggregation**: A "has-an" instance of B. B can exist without A.
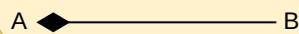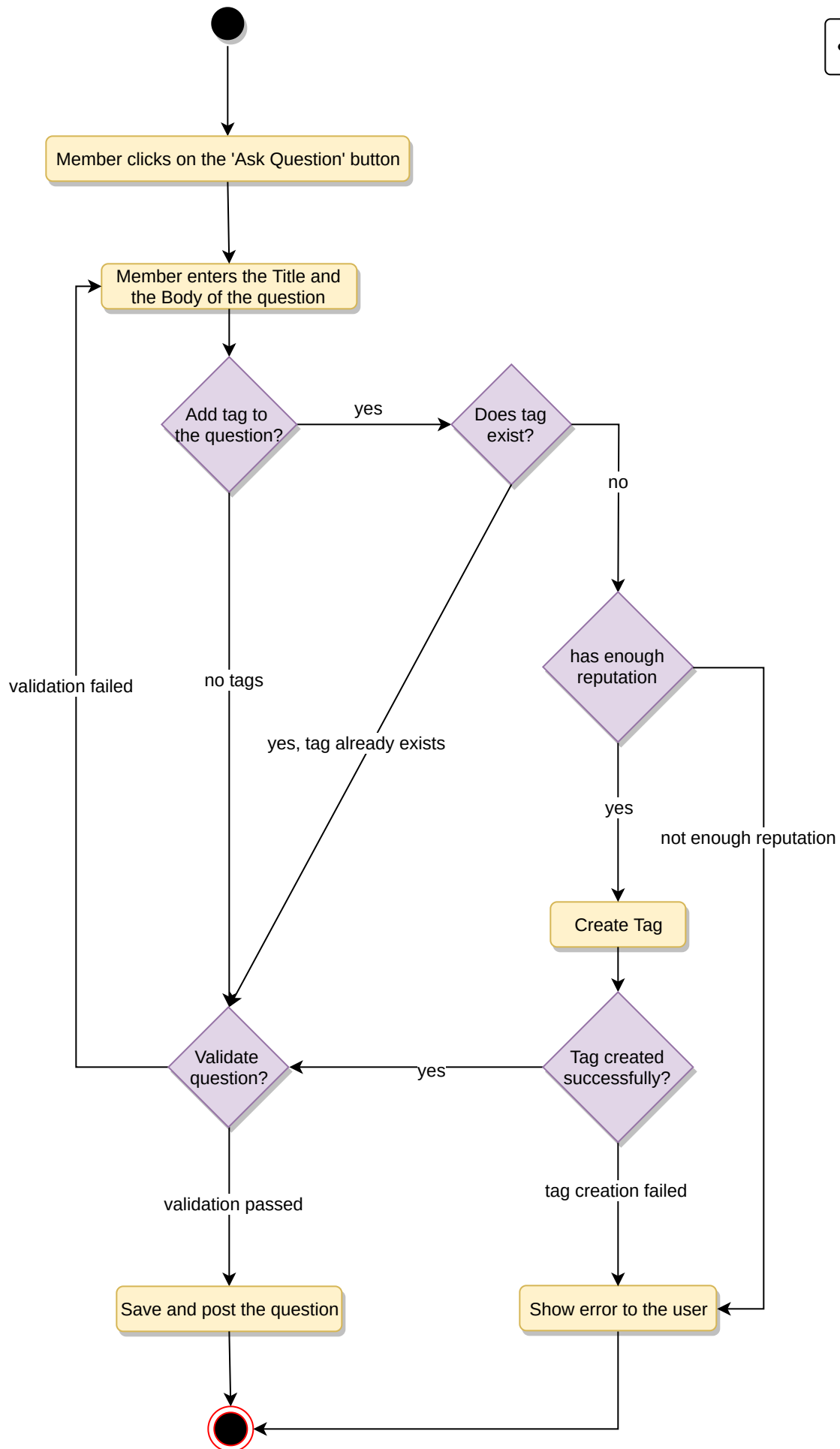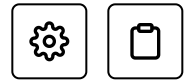
A ◆———————— B   **Composition**: A "has-an" instance of B. B cannot exist without A.

Activity diagrams
#

Post a new question: Any member or moderator can perform this activity. Here are the steps to post a question:

Member clicks on the 'Ask Question' button

Member enters the Title and the Body of the question

Add tag to the question?

yes

Does tag exist?

no

has enough reputation

validation failed

no tags

yes, tag already exists

not enough reputation

yes

Create Tag

Validate question?

yes

Tag created successfully?

validation passed

tag creation failed

Save and post the question
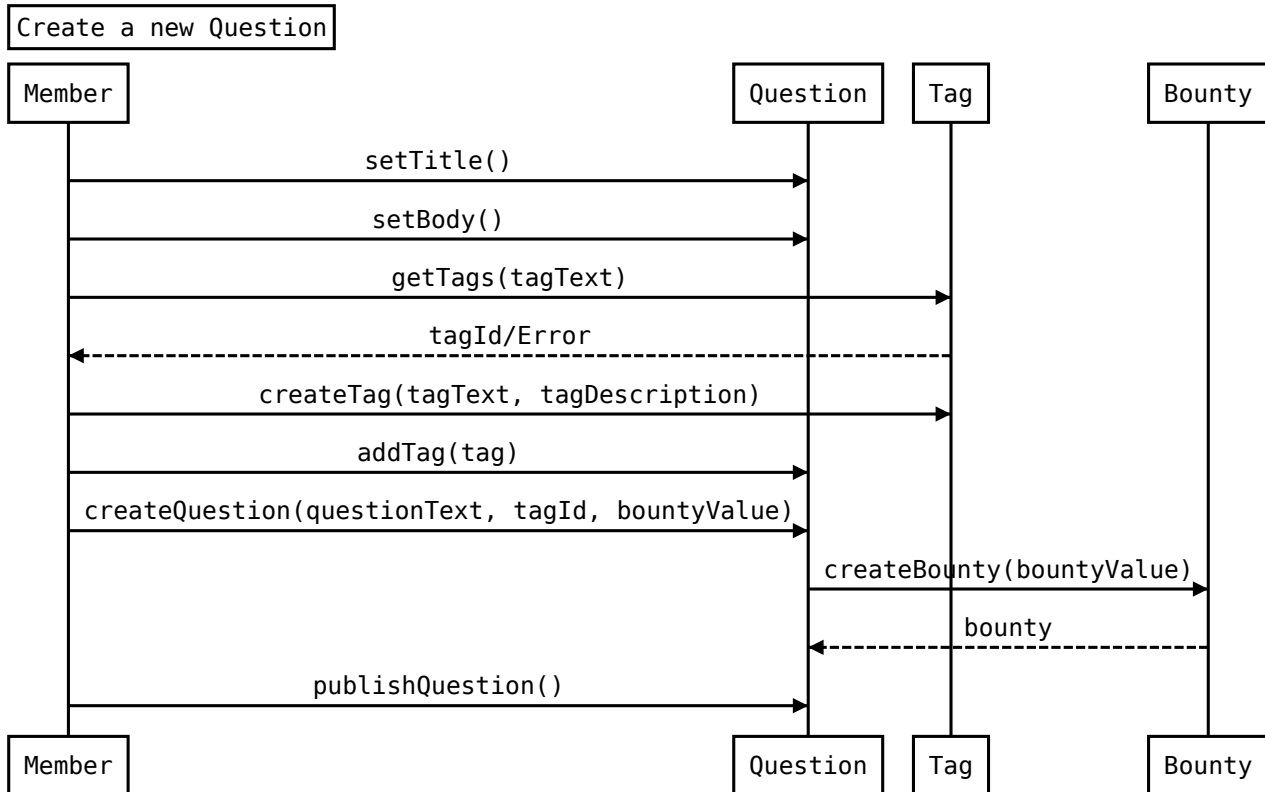
Show error to the user

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:
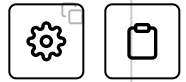
☕ Java	🐍 Python

```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

```
// For simplicity, we are not defining getter and setter functions. The reader can
// assume that all class attributes are private and accessed through their respecti
// 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 {
  public boolean blockMember(Member member);
  public boolean unblockMember(Member member);
}

public class Moderator extends Member {
  public boolean closeQuestion(Question question);
  public boolean undeleteQuestion(Question question);
}
```

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();
}
```
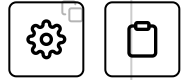
Photo and Bounty: Members can put bounties on questions. Answers and Questions can have multiple photos:

☕ 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 descripti
  }
}

public class Comment {
  private String text;
  private Date creationTime;
  private int flagCount;
  private int voteCount;

  private Member askingMember;

  public boolean incrementVoteCount();
}

public class Answer {
  private String answerText;
  private boolean accepted;
  private int voteCount;
  private int flagCount;
  private Date creationTime;

  private Member creatingMember;
  private List<Photo> photos;

  public boolean incrementVoteCount();
}
```

⚙️   📋

← **Back**   **Next** →

Design Amazon - Online Shopping Sy...   Design a Movie Ticket Booking System

☑️ **Completed**

12% completed, meet the <u>criteria</u> and claim your course certificate!

🔒

⊘ Report
    an Issue

❓ Ask a Question
(https://discuss.educative.io/tag/design-stack-overflow__object-oriented-design-case-
studies__grokking-the-object-oriented-design-interview)