36. Stack OverFlow

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.

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.

Stack Overflow

- Stack Overflow - Online Community for Developers

System Requirements

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:

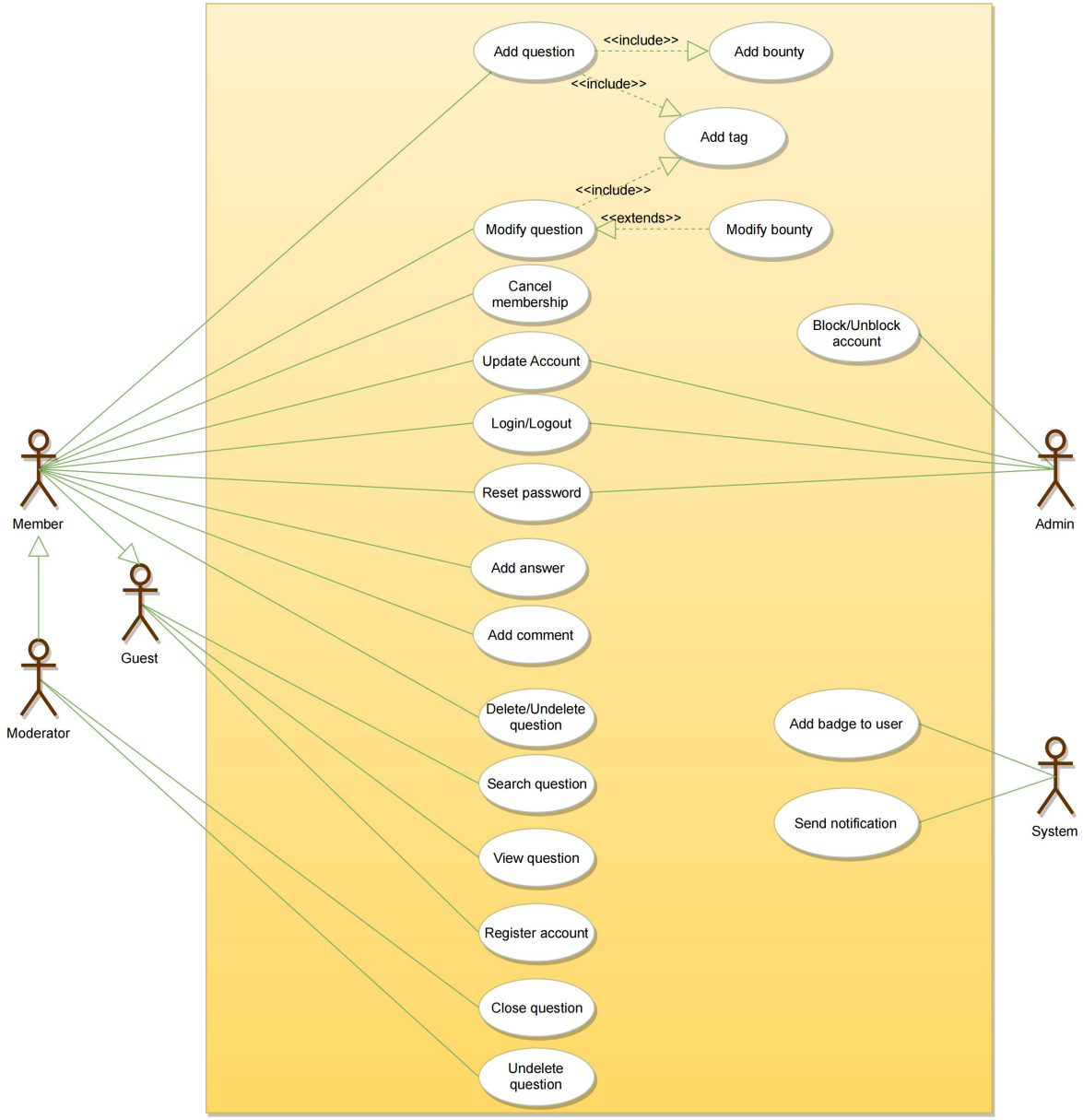
- **Admin**: Mainly responsible for blocking or unblocking members.
- **Guest**: All guests can search and view questions.
- **Member**: Members can perform all activities that guests can, in addition to which they can add/remove questions, answers, and comments. Members can delete and un-delete their 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.

Here is the use case diagram of Stack Overflow:

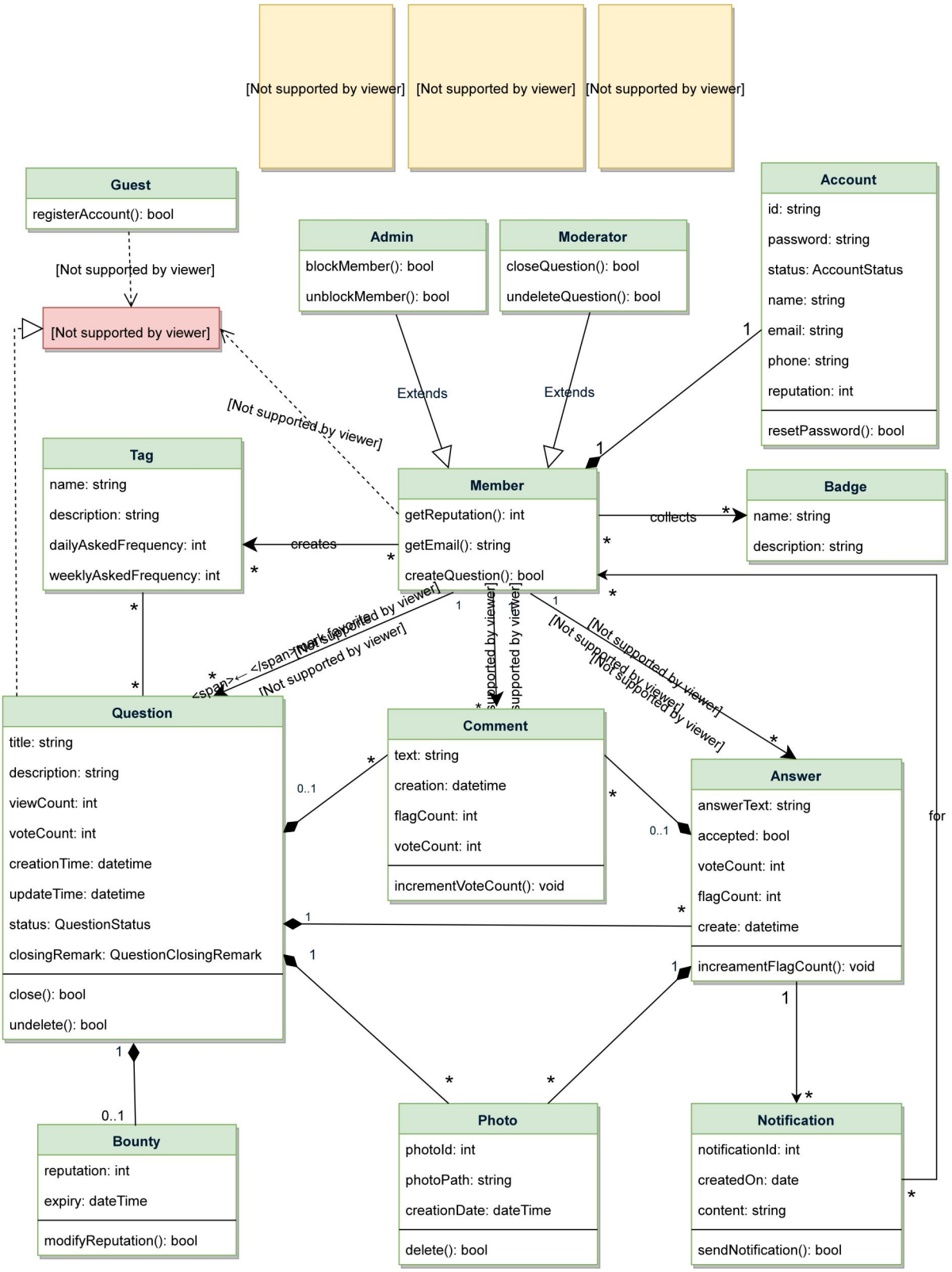


Use Case Diagram for Stack Overflow

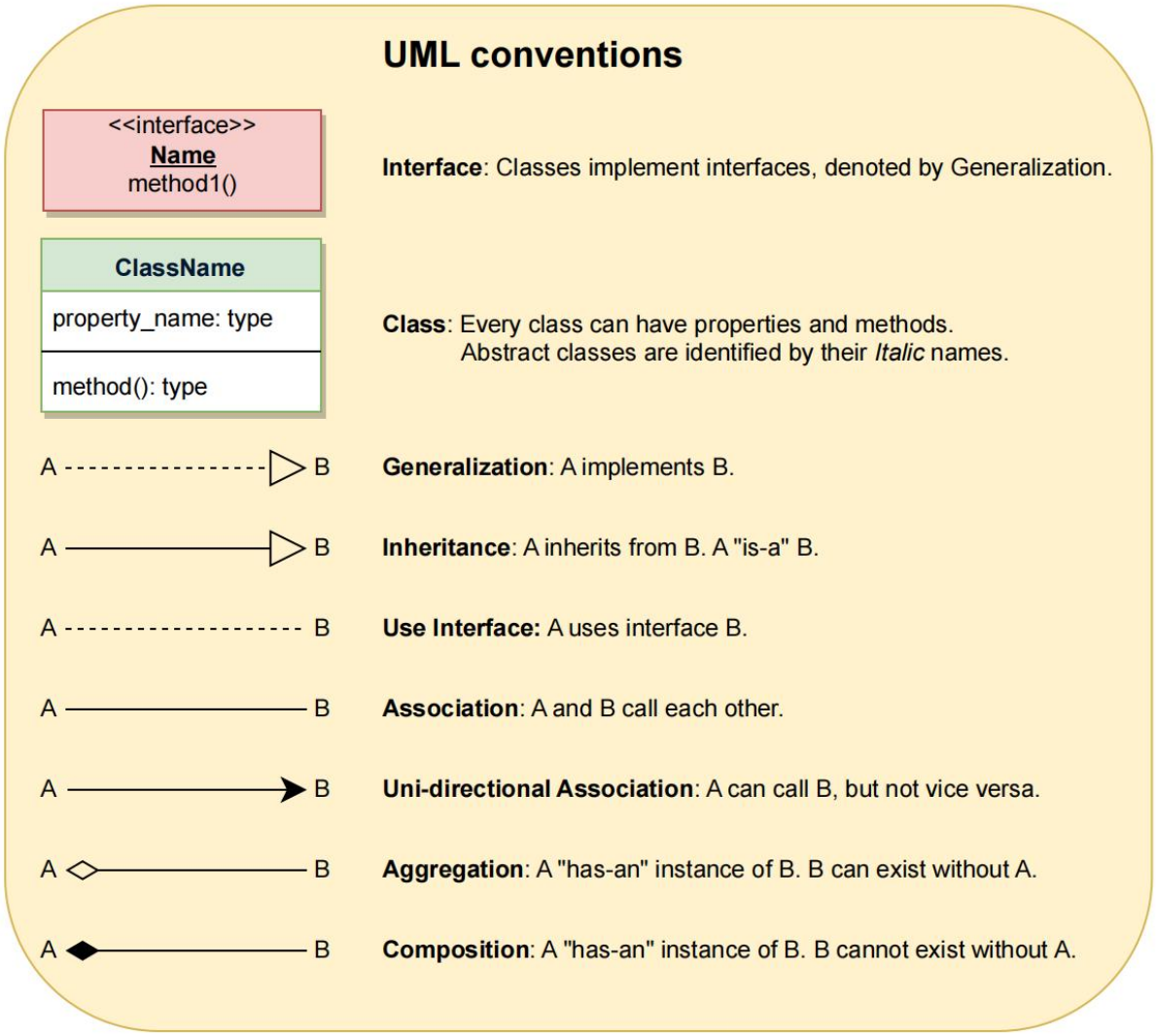
Class Diagram

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.



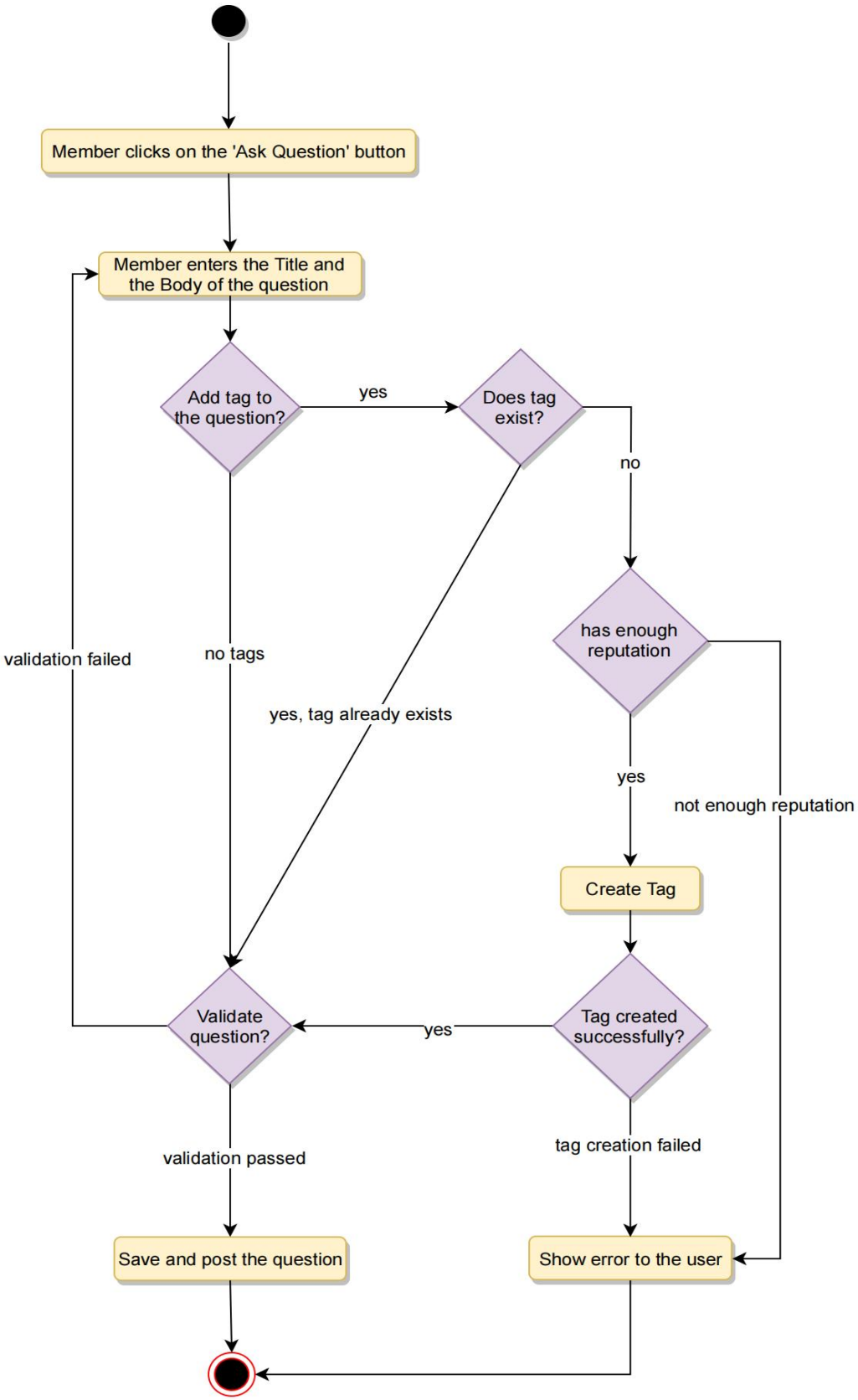
Class Diagram for Stack Overflow



UML for Stack Overflow

Activity Diagram

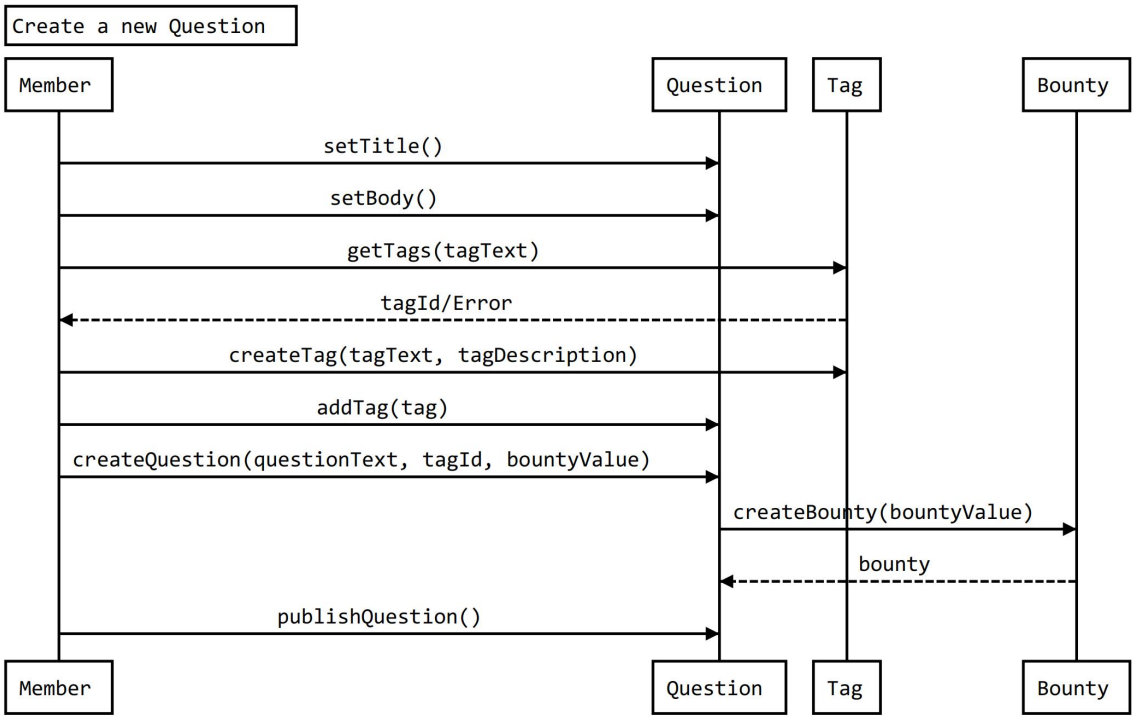
Post a new question: Any member or moderator can perform this activity. Here are the steps to post a question:



Activity Diagram for Stack Overflow

Sequence Diagram

Following is the sequence diagram for creating a new question:



Sequence Diagram for Stack Overflow

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:

1. Enums for Statuses

```
public enum QuestionStatus {
    OPEN(1), CLOSED(2), ON_HOLD(3), DELETED(4);

    private final int status;

    QuestionStatus(int status) {
        this.status = status;
    }

    public int getStatus() {
        return status;
    }
}

public enum QuestionClosingRemark {
    DUPLICATE(1), OFF_TOPIC(2), TOO_BROAD(3), NOT_CONSTRUCTIVE(4),
    NOT_A_REAL_QUESTION(5), PRIMARILY_OPINION_BASED(6);

    private final int remark;

    QuestionClosingRemark(int remark) {
        this.remark = remark;
    }

    public int getRemark() {
        return remark;
```

```
    }
}

public enum AccountStatus {
    ACTIVE(1), CLOSED(2), CANCELED(3), BLACKLISTED(4), BLOCKED(5);

    private final int status;

    AccountStatus(int status) {
        this.status = status;
    }

    public int getStatus() {
        return status;
    }
}
```

2. Account, Member, Admin, Moderator Classes

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

    public Account(String id, String password, String name, String address, String email, String phone, AccountStatus status) {
        this.id = id;
        this.password = password;
```

```
            this.name = name;
            this.address = address;
            this.email = email;
            this.phone = phone;
            this.status = status != null ? status : AccountStatus.ACTIVE;
            this.reputation = 0;
        }

        public void resetPassword() {
            // Implementation to reset password
        }

        public int getReputation() {
            return reputation;
        }

        public String getEmail() {
            return email;
        }
}

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

        public Member(Account account) {
            this.account = account;
            this.badges = new ArrayList<>();
        }

        public int getReputation() {
            return account.getReputation();
        }

        public String getEmail() {
```

```
            return account.getEmail();
        }

        public void createQuestion(Question question) {
            // Implementation to create a question
        }

        public void createTag(Tag tag) {
            // Implementation to create a tag
        }
}

class Admin extends Member {
        public Admin(Account account) {
            super(account);
        }

        public void blockMember(Member member) {
            // Implementation to block a member
        }

        public void unblockMember(Member member) {
            // Implementation to unblock a member
        }
}

class Moderator extends Member {
        public Moderator(Account account) {
            super(account);
        }

        public void closeQuestion(Question question) {
            // Implementation to close a question
        }
```

```
    public void undeleteQuestion(Question question) {
        // Implementation to undelete a question
    }
}
```

3. Badge, Tag, Notification Classes

```
class Badge {
    private String name;
    private String description;

    public Badge(String name, String description) {
        this.name = name;
        this.description = description;
    }
}

class Tag {
    private String name;
    private String description;
    private int dailyAskedFrequency;
    private int weeklyAskedFrequency;

    public Tag(String name, String description) {
        this.name = name;
        this.description = description;
        this.dailyAskedFrequency = 0;
        this.weeklyAskedFrequency = 0;
    }
}

import java.time.LocalDateTime;

class Notification {
```

```
    private String notificationId;
    private LocalDateTime createdOn;
    private String content;

    public Notification(String notificationId, String content) {
        this.notificationId = notificationId;
        this.createdOn = LocalDateTime.now();
        this.content = content;
    }

    public void sendNotification() {
        // Implementation to send notification
    }
}
```

4. Photo and Bounty Classes

```
import java.time.LocalDateTime;

class Photo {
    private String photoId;
    private String photoPath;
    private LocalDateTime creationDate;
    private Member creatingMember;

    public Photo(String photoId, String photoPath, Member creatingMember) {
        this.photoId = photoId;
        this.photoPath = photoPath;
        this.creationDate = LocalDateTime.now();
        this.creatingMember = creatingMember;
    }

    public void delete() {
        // Implementation to delete photo
```

```
    }
}

class Bounty {
    private int reputation;
    private LocalDateTime expiry;

    public Bounty(int reputation, LocalDateTime expiry) {
        this.reputation = reputation;
        this.expiry = expiry;
    }

    public void modifyReputation(int reputation) {
        // Modify the reputation
        this.reputation = reputation;
    }
}
```

5. Question, Comment, Answer, and Search Classes

```
import java.time.LocalDateTime;
import java.util.List;

abstract class Search {
    public abstract List<Question> search(String query);
}

class Question extends Search {
    private String title;
    private String description;
    private int viewCount;
    private int voteCount;
    private LocalDateTime creationTime;
    private LocalDateTime updateTime;
```

```
    private QuestionStatus status;
    private QuestionClosingRemark closingRemark;
    private Bounty bounty;
    private Member askingMember;
    private List<Photo> photos;
    private List<Comment> comments;
    private List<Answer> answers;

    public Question(String title, String description, Bounty bounty, Member askingMember) {
        this.title = title;
        this.description = description;
        this.viewCount = 0;
        this.voteCount = 0;
        this.creationTime = LocalDateTime.now();
        this.updateTime = LocalDateTime.now();
        this.status = QuestionStatus.OPEN;
        this.closingRemark = QuestionClosingRemark.DUPLICATE;
        this.bounty = bounty;
        this.askingMember = askingMember;
        this.photos = new ArrayList<>();
        this.comments = new ArrayList<>();
        this.answers = new ArrayList<>();
    }

    public void close() {
        this.status = QuestionStatus.CLOSED;
    }

    public void undelete() {
        // Implementation to undelete question
    }

    public void addComment(Comment comment) {
        comments.add(comment);
    }
```

```
    public void addBounty(Bounty bounty) {
        this.bounty = bounty;
    }

    @Override
    public List<Question> search(String query) {
        // Implementation to search questions based on the query
        return null; // Placeholder
    }
}

class Comment {
    private String text;
    private LocalDateTime creationTime;
    private int flagCount;
    private int voteCount;
    private Member askingMember;

    public Comment(String text, Member askingMember) {
        this.text = text;
        this.creationTime = LocalDateTime.now();
        this.flagCount = 0;
        this.voteCount = 0;
        this.askingMember = askingMember;
    }

    public void incrementVoteCount() {
        voteCount++;
    }
}

class Answer {
    private String answerText;
    private boolean accepted;
```

```
    private int voteCount;
    private int flagCount;
    private LocalDateTime creationTime;
    private Member creatingMember;
    private List<Photo> photos;

    public Answer(String answerText, Member creatingMember) {
        this.answerText = answerText;
        this.accepted = false;
        this.voteCount = 0;
        this.flagCount = 0;
        this.creationTime = LocalDateTime.now();
        this.creatingMember = creatingMember;
        this.photos = new ArrayList<>();
    }

    public void incrementVoteCount() {
        voteCount++;
    }
}
```