Designing Stack Overflow

→ StackOverflow is a Q & A website for programmers & developers

→ Registered users can post new questions & answers questions from other users.

→ Each user can collect reputation points.

→ points are affected by upvotes & downvotes.

→ More Reputation point allow users to perform additional functions like voting to close or delete a question.

→ Users are awarded badges to highlight their Credibility.

Expectation from the interviewee

→ How are users able to search for questions?

→ Is there a way to filter questions using tags or users?

→ How are reputation points calculated? Do users get points for asking or answering question?

→ How many points are required for users to get a moderator access.

→ What are the different types of voting allowed on Stack Overflow? Are you allowed to upvote & downvote.

→ How does voting works a question has to be closed & deleted? which user can vote in such circumstances.

→ How are reputation points awarded on bounty questions?

→ When do users start start a bounty? How long does a bounty last before expiring.

Requirement Collection

R1: Any guest can view questions & search question by tag, username or words.

R2: Users should be able to post new questions & add answers to an open questions

R3: Users can flag a question, answer, or comment if anything goes against the community guidelines.

R4: A user can upvote, downvote & add comments to a question or answer, while they can only upvote a comment.

R5: Users can vote a delete or vote to close off questions for community-specific reasons. However, they can only vote to delete an answer.

R6: Any user can add a bounty to their questions to attract more answers.

R7: Moderators can close a question or restore an already deleted questions. Moderators can also delete answers.

R8: The system should send the user a notification whenever there has been an interaction with them, such as the user's question receiving an answer, earning a badge, or someone upvoting or downvoting their posts.

R9: Users can earn badge for their helpful answers or comments.

R10: The system should also be able to determine the most popular tags used in questions

R11: Users can add tag to their questions. A tag is a word or phase that describe the topic of the question.

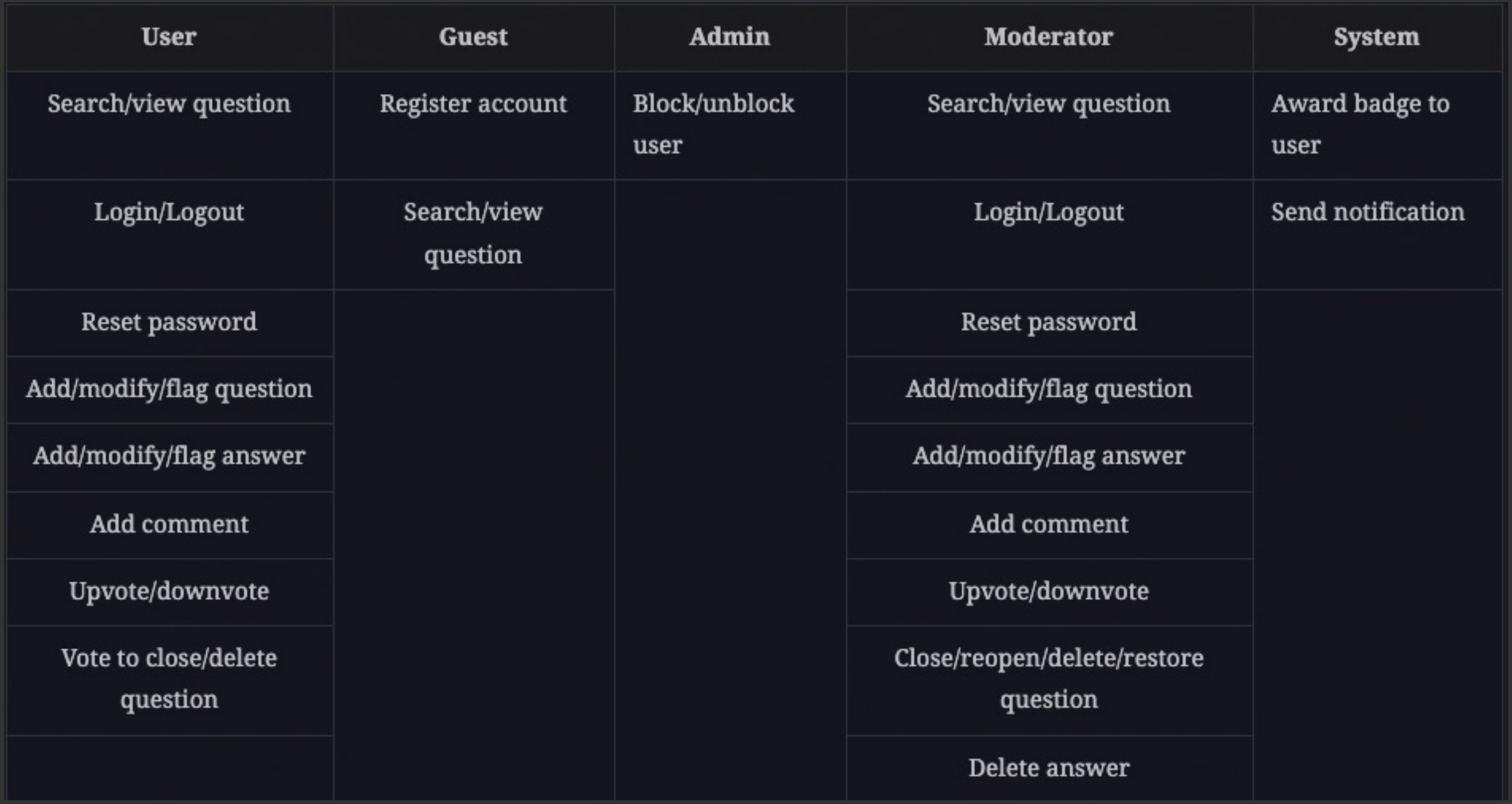
Use Case Diagram

Primary Actors

- User

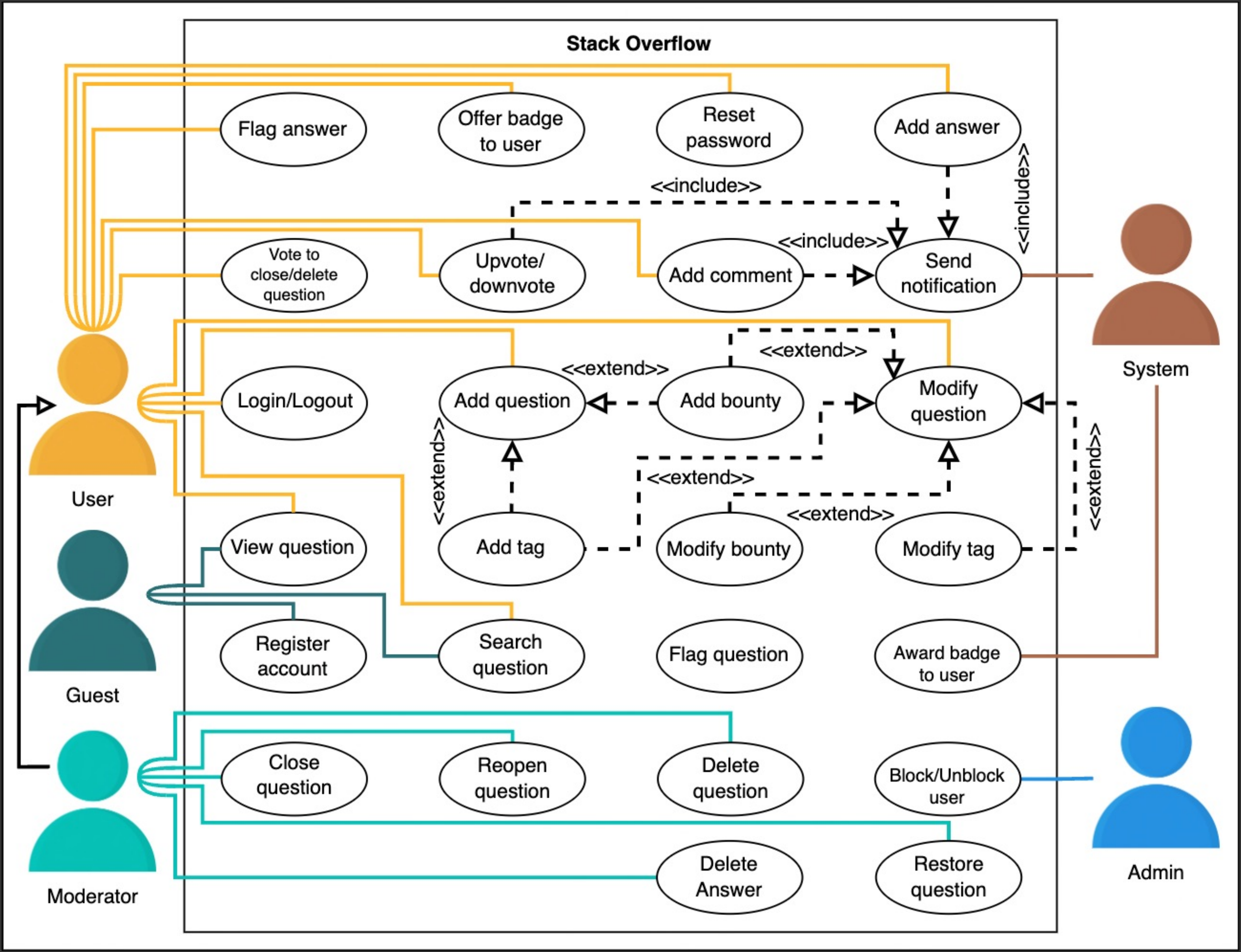
Secondary Actors

- Guests
- Admin
- Moderator
- System



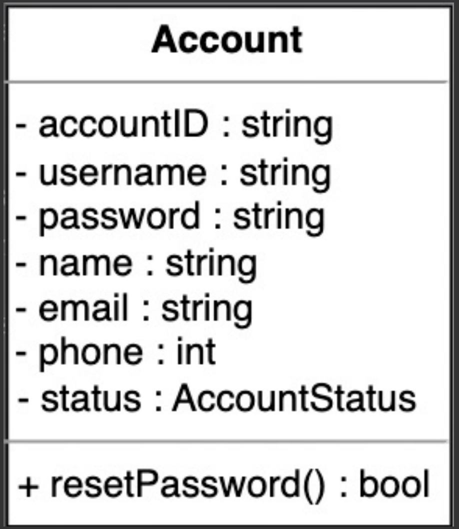
User	Guest	Admin	Moderator	System
Search/view question	Register account	Block/unblock user	Search/view question	Award badge to user
Login/Logout	Search/view question		Login/Logout	Send notification
Reset password			Reset password	
Add/modify/flag question			Add/modify/flag question	
Add/modify/flag answer			Add/modify/flag answer	
Add comment			Add comment	
Upvote/downvote			Upvote/downvote	
Vote to close/delete question			Close/reopen/delete/restore question	
			Delete answer	

Use Case Diagram



Class Diagram

1. Account

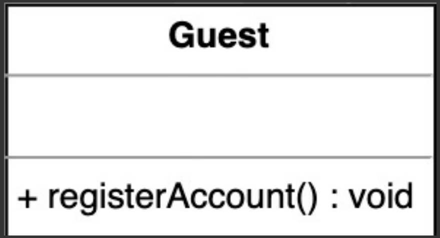


Account

- accountID : string
- username : string
- password : string
- name : string
- email : string
- phone : int
- status : AccountStatus

+ resetPassword() : bool

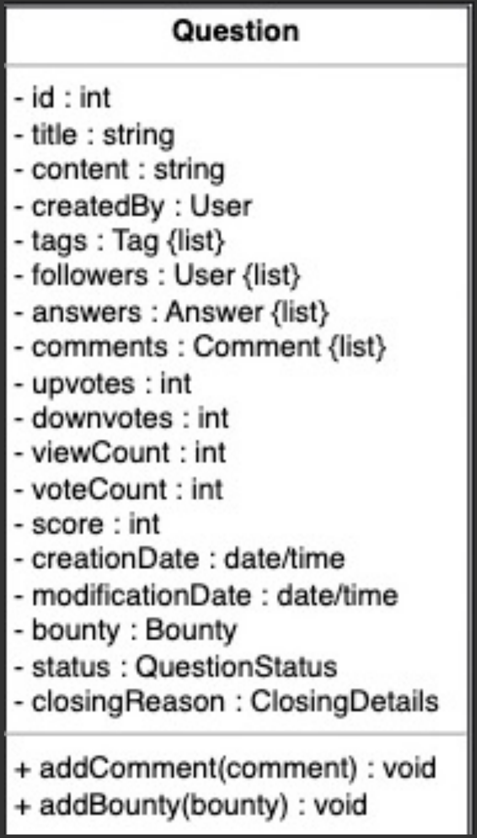
2. Guests



Guest

+ registerAccount() : void

3. Questions

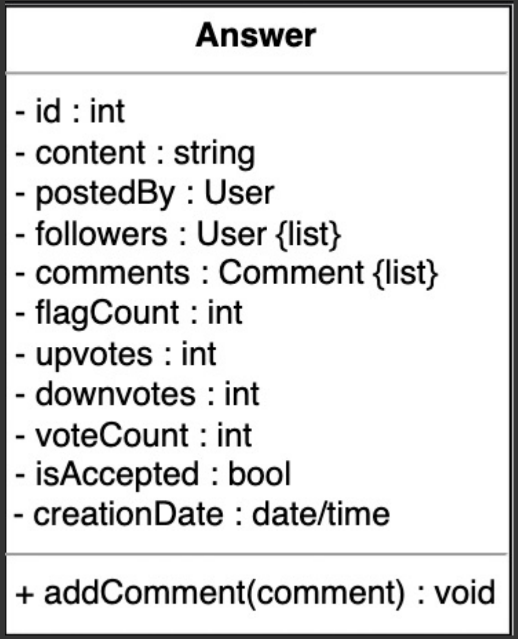


Question

- id : int
- title : string
- content : string
- createdBy : User
- tags : Tag {list}
- followers : User {list}
- answers : Answer {list}
- comments : Comment {list}
- upvotes : int
- downvotes : int
- viewCount : int
- voteCount : int
- score : int
- creationDate : date/time
- modificationDate : date/time
- bounty : Bounty
- status : QuestionStatus
- closingReason : ClosingDetails

+ addComment(comment) : void
+ addBounty(bounty) : void

4. Answer

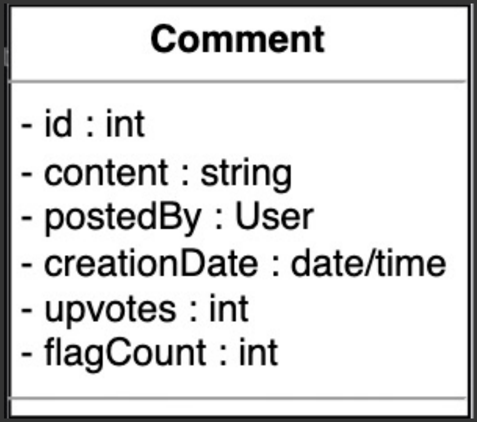


Answer

- id : int
- content : string
- postedBy : User
- followers : User {list}
- comments : Comment {list}
- flagCount : int
- upvotes : int
- downvotes : int
- voteCount : int
- isAccepted : bool
- creationDate : date/time

+ addComment(comment) : void

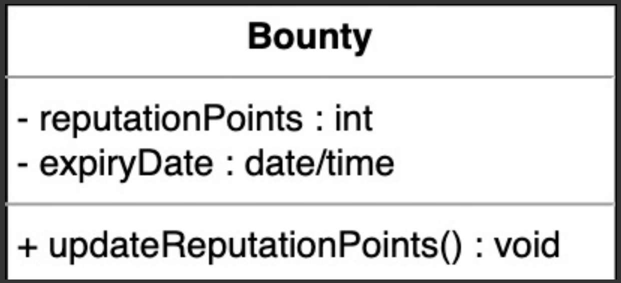
5. Comment



Comment

- id : int
- content : string
- postedBy : User
- creationDate : date/time
- upvotes : int
- flagCount : int

6. Bounty

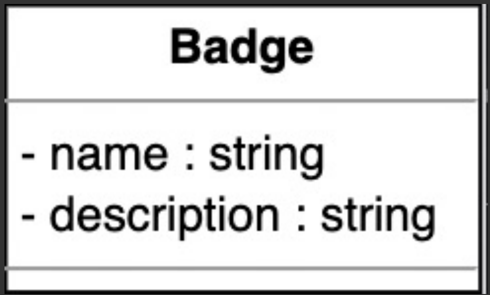


Bounty

- reputationPoints : int
- expiryDate : date/time

+ updateReputationPoints() : void

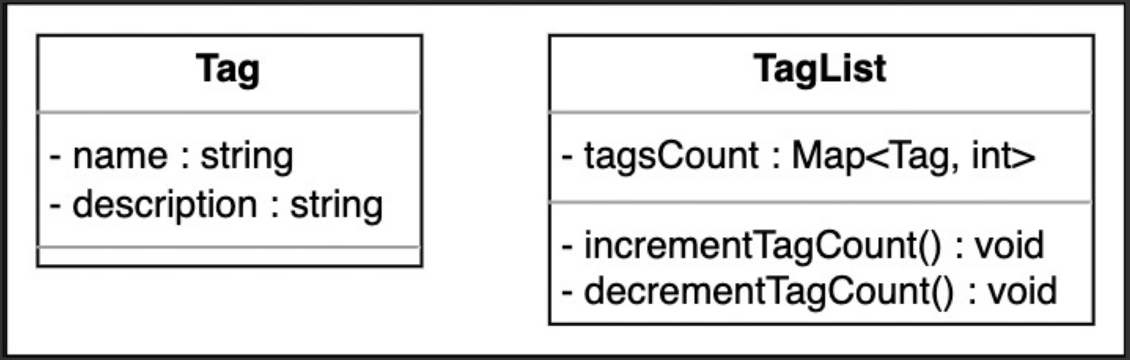
7. Badge



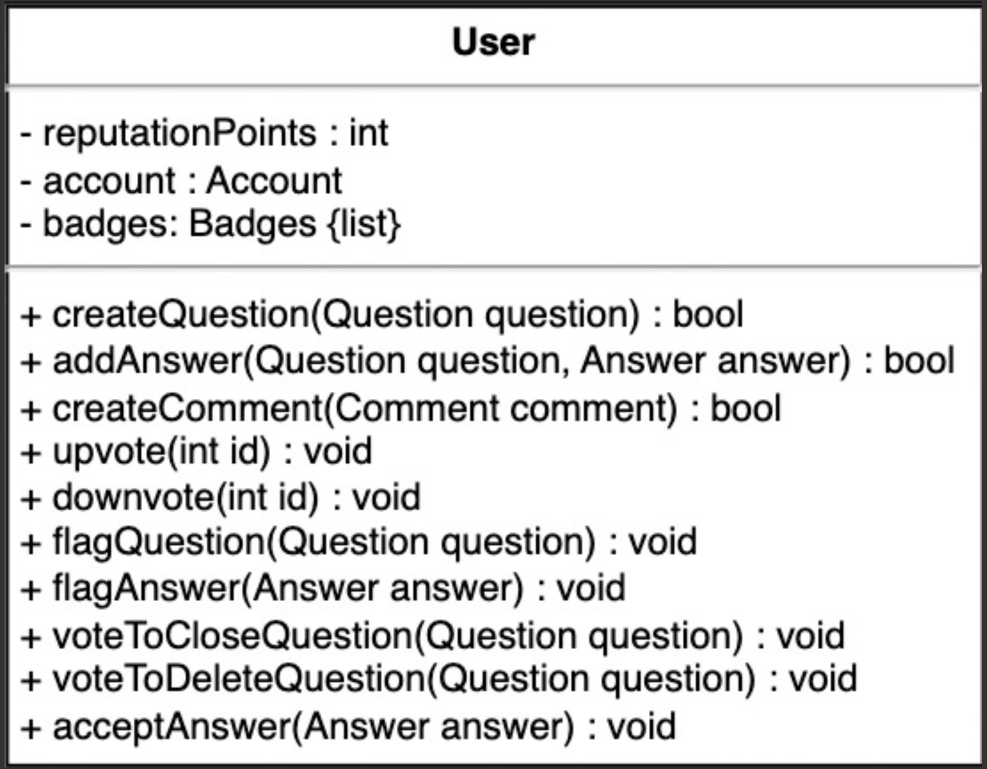
Badge

- name : string
- description : string

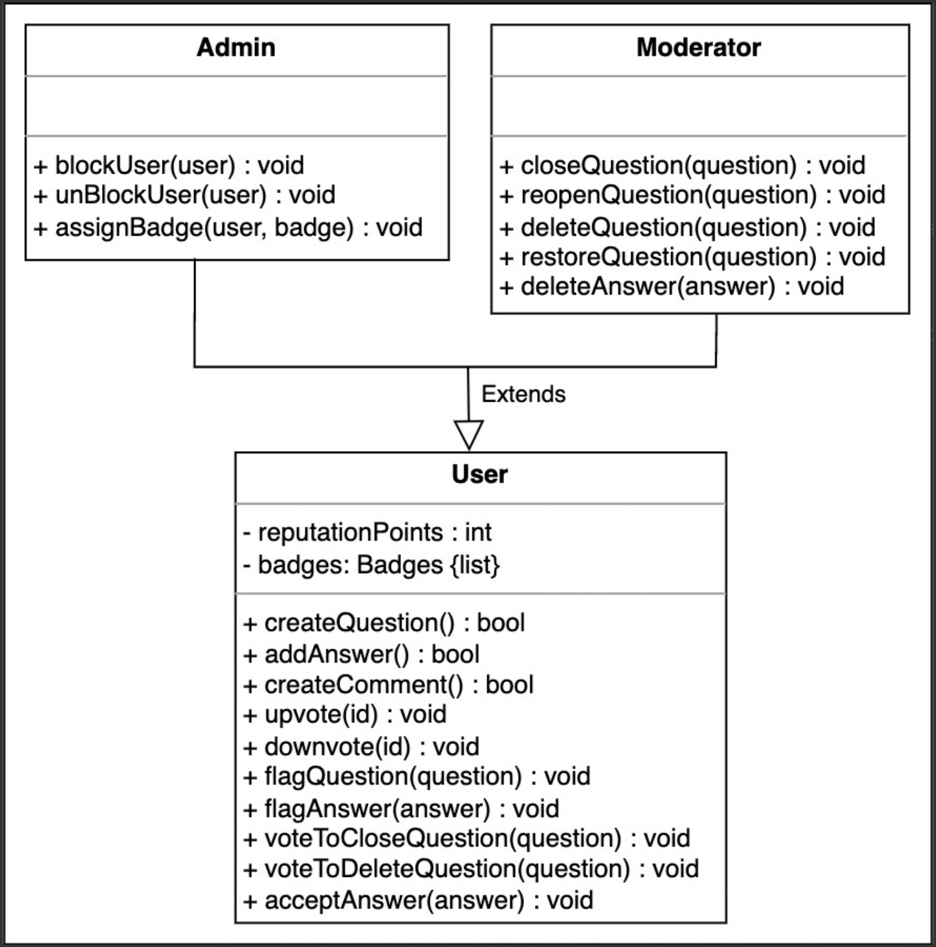
8. Tag & Taglist


Tag
- name : string
- description : string
TagList
- tagsCount : Map<Tag, int>
- incrementTagCount() : void
- decrementTagCount() : void

9. User

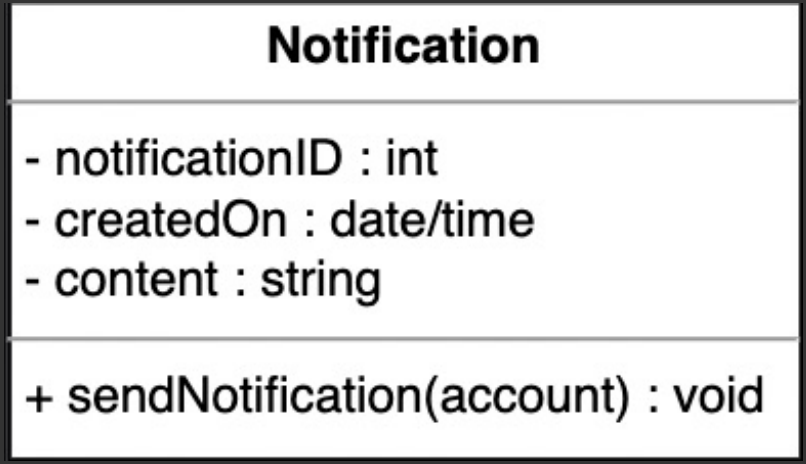

User
- reputationPoints : int
- account : Account
- badges: Badges {list}
+ createQuestion(Question question) : bool
+ addAnswer(Question question, Answer answer) : bool
+ createComment(Comment comment) : bool
+ upvote(int id) : void
+ downvote(int id) : void
+ flagQuestion(Question question) : void
+ flagAnswer(Answer answer) : void
+ voteToCloseQuestion(Question question) : void
+ voteToDeleteQuestion(Question question) : void
+ acceptAnswer(Answer answer) : void

10 Admin & Moderator

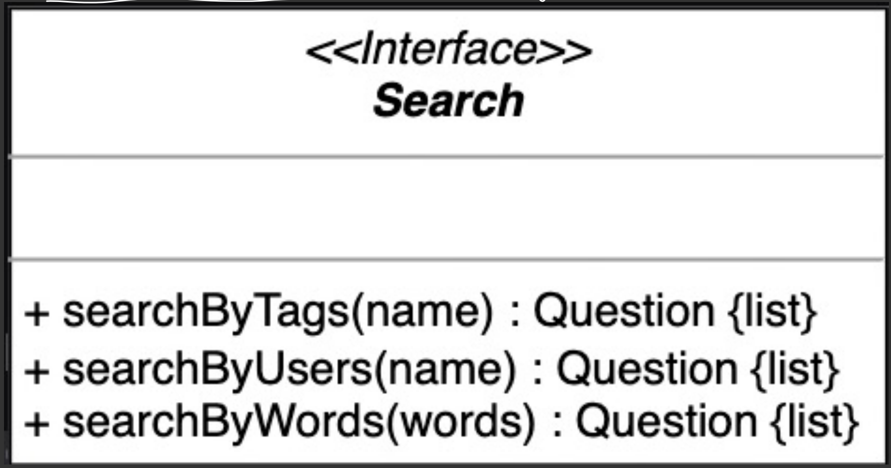


Admin
+ blockUser(user) : void
+ unBlockUser(user) : void
+ assignBadge(user, badge) : void
Moderator
+ closeQuestion(question) : void
+ reopenQuestion(question) : void
+ deleteQuestion(question) : void
+ restoreQuestion(question) : void
+ deleteAnswer(answer) : void
Extends
User
- reputationPoints : int
- badges: Badges {list}
+ createQuestion() : bool
+ addAnswer() : bool
+ createComment() : bool
+ upvote(id) : void
+ downvote(id) : void
+ flagQuestion(question) : void
+ flagAnswer(answer) : void
+ voteToCloseQuestion(question) : void
+ voteToDeleteQuestion(question) : void
+ acceptAnswer(answer) : void

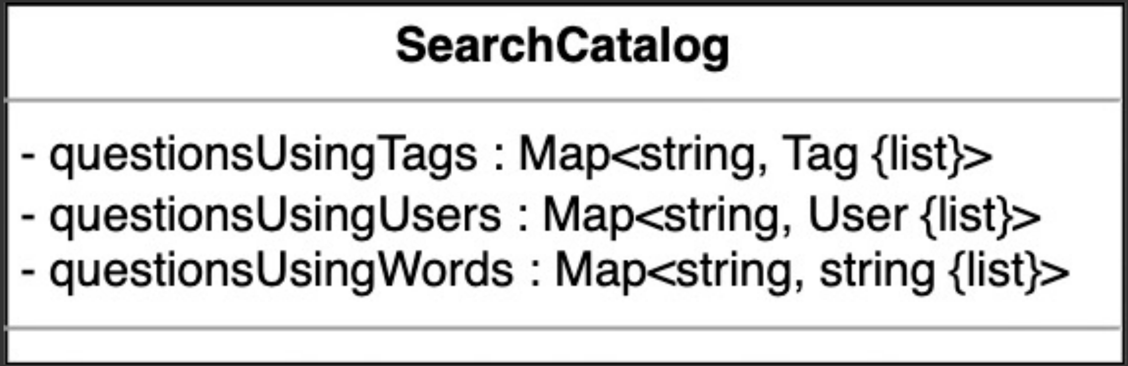
11. Notification



12. Search & Search Catlog

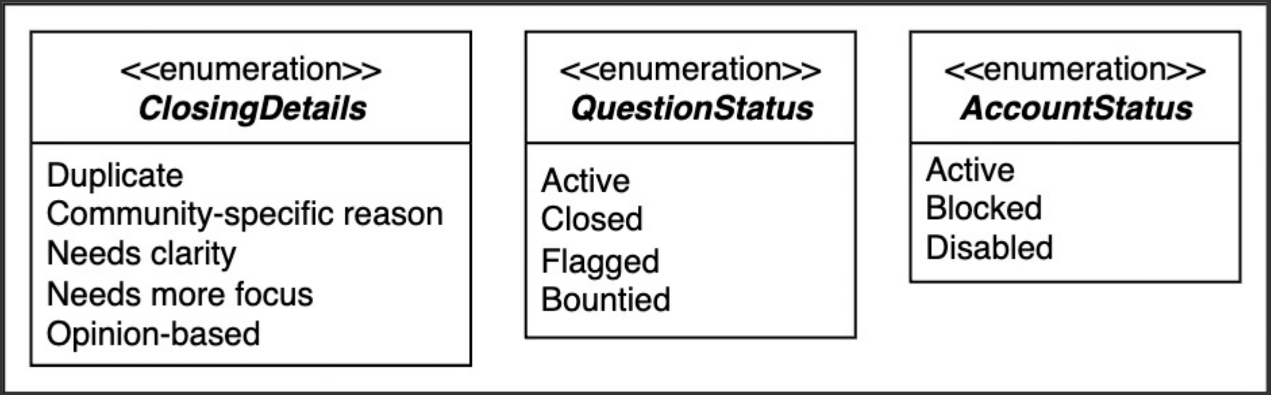


<<Interface>>
Search
+ searchByTags(name) : Question {list}
+ searchByUsers(name) : Question {list}
+ searchByWords(words) : Question {list}

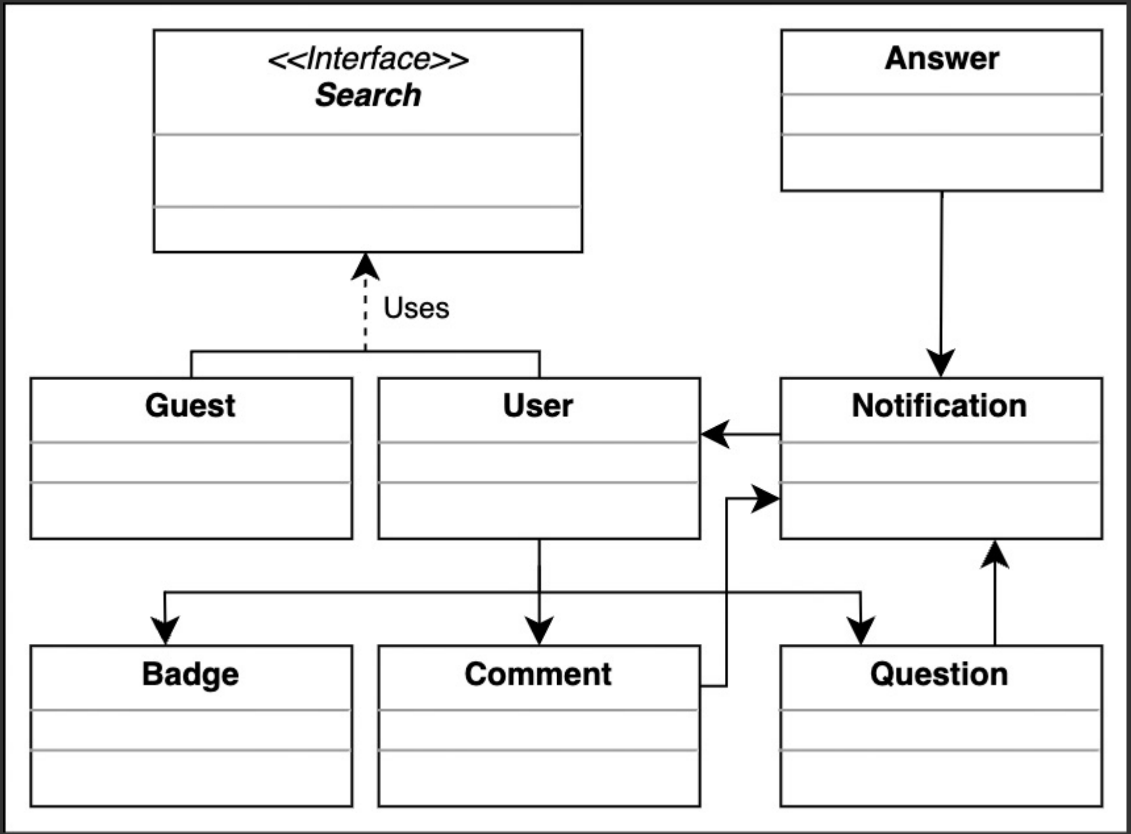


SearchCatalog
- questionsUsingTags : Map<string, Tag {list}>
- questionsUsingUsers : Map<string, User {list}>
- questionsUsingWords : Map<string, string {list}>

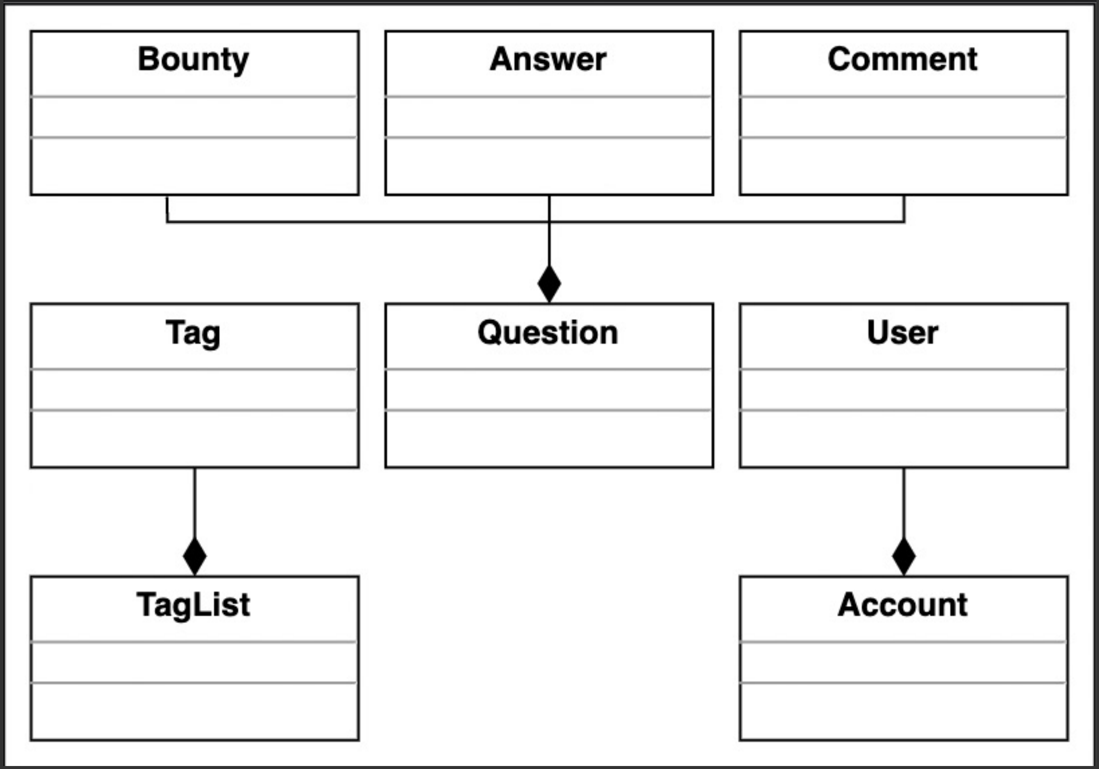
Enumerations



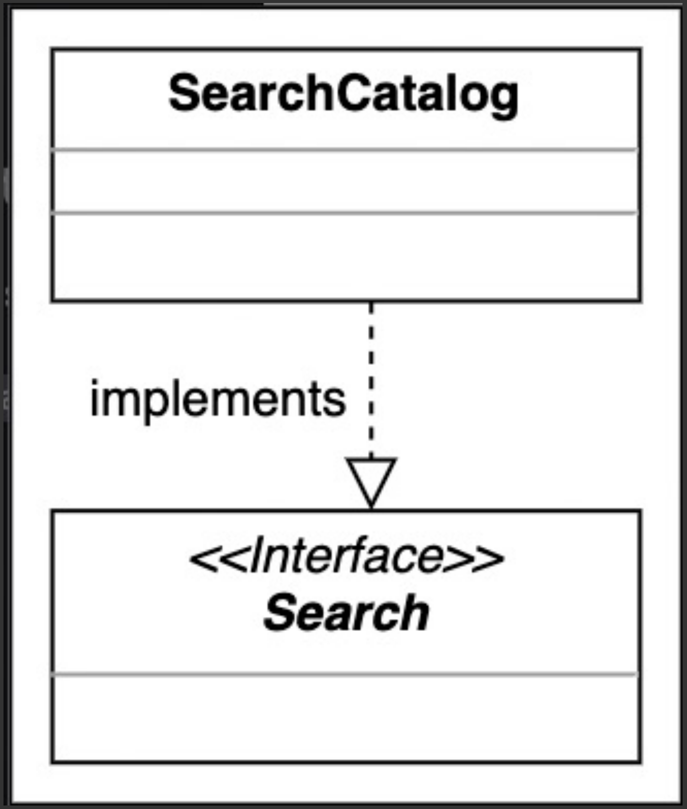
→ Associations



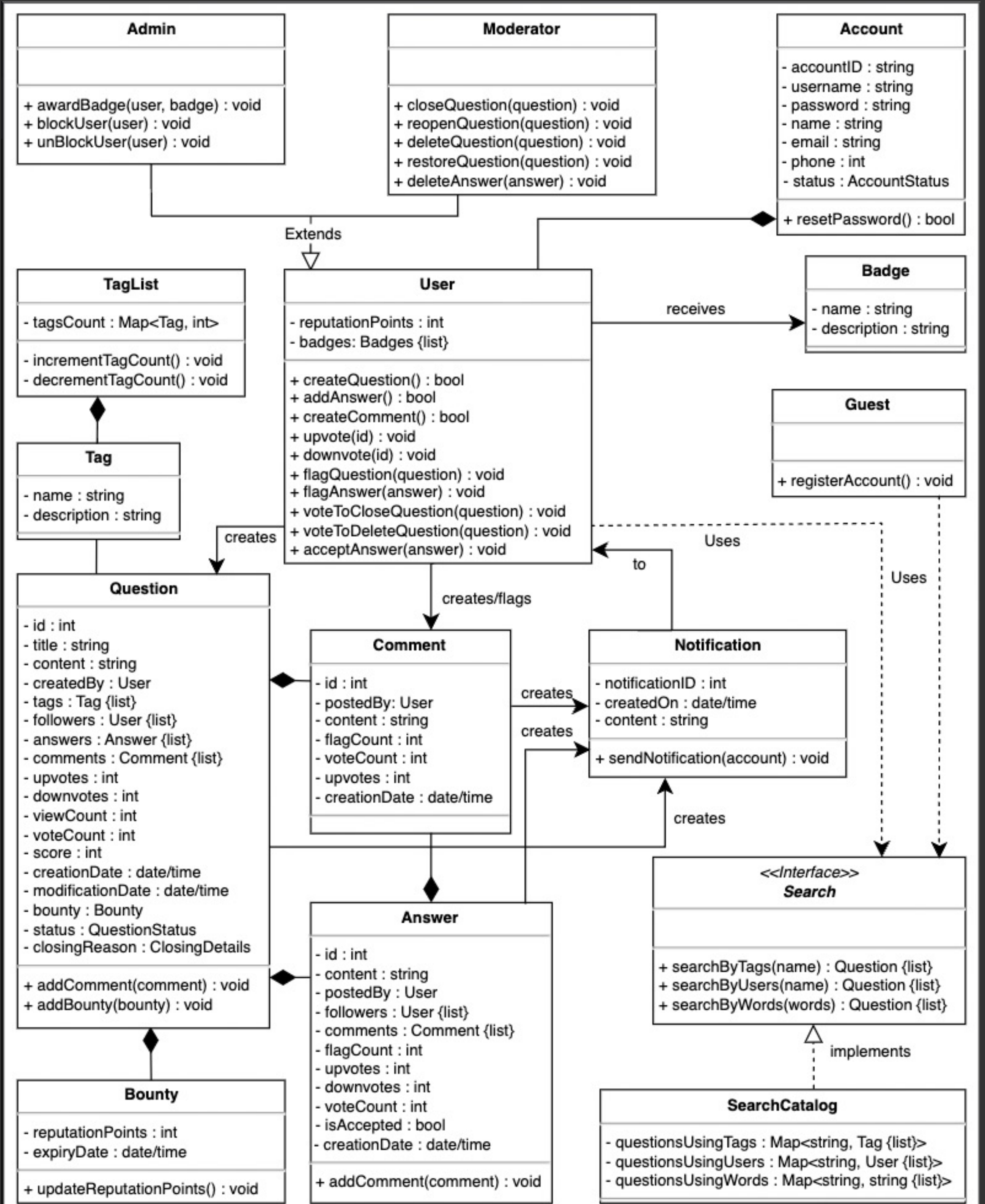
$\rightarrow$ Composition



Generalization

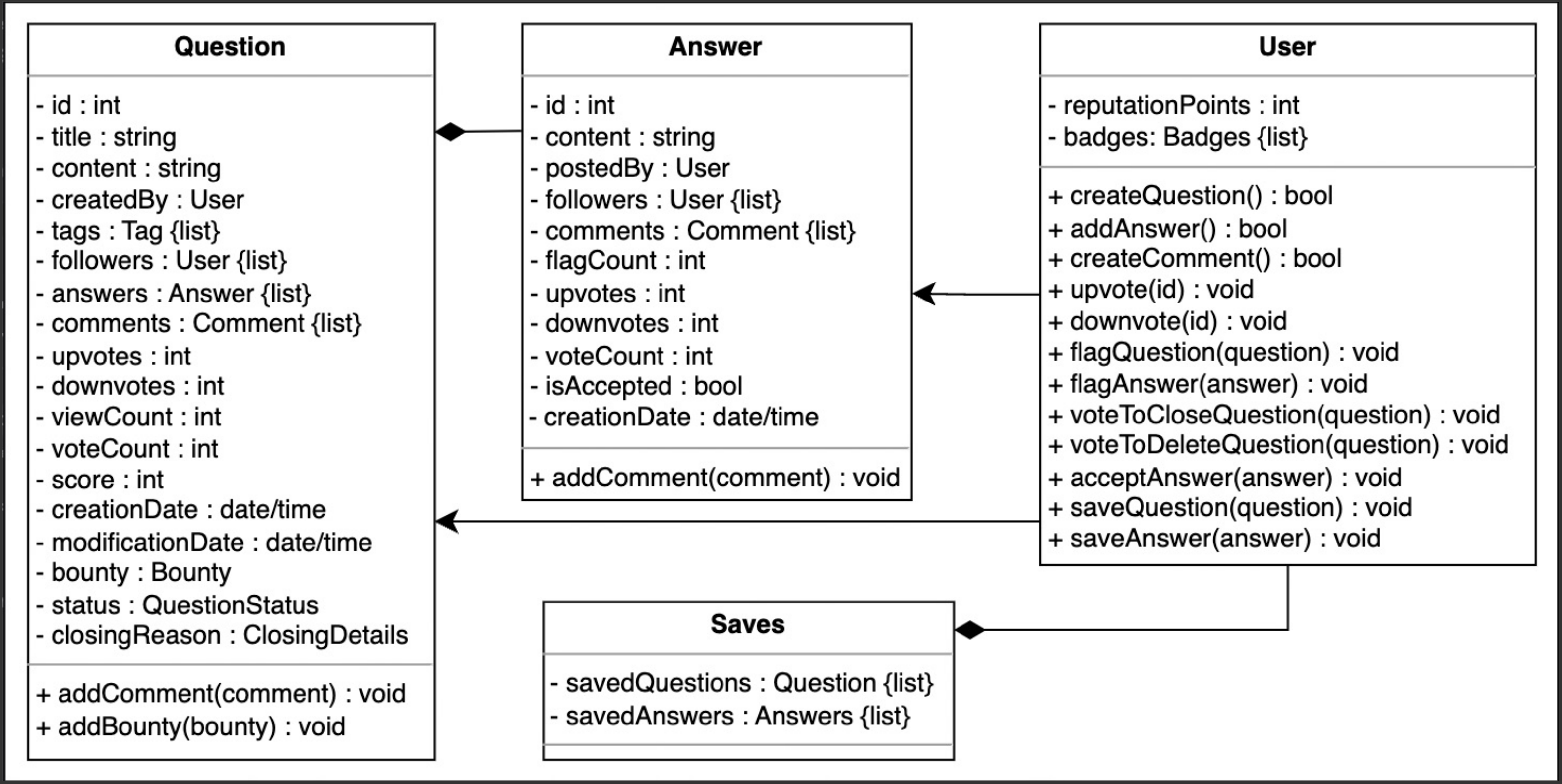


Class Diagram $\longrightarrow$


Admin
+ awardBadge(user, badge) : void
+ blockUser(user) : void
+ unBlockUser(user) : void
Moderator
+ closeQuestion(question) : void
+ reopenQuestion(question) : void
+ deleteQuestion(question) : void
+ restoreQuestion(question) : void
+ deleteAnswer(answer) : void
Account
- accountID : string
- username : string
- password : string
- name : string
- email : string
- phone : int
- status : AccountStatus
+ resetPassword() : bool
Extends
TagList
- tagsCount : Map<Tag, int>
- incrementTagCount() : void
- decrementTagCount() : void
User
- reputationPoints : int
- badges: Badges {list}
+ createQuestion() : bool
+ addAnswer() : bool
+ createComment() : bool
+ upvote(id) : void
+ downvote(id) : void
+ flagQuestion(question) : void
+ flagAnswer(answer) : void
+ voteToCloseQuestion(question) : void
+ voteToDeleteQuestion(question) : void
+ acceptAnswer(answer) : void
receives
Badge
- name : string
- description : string
Guest
+ registerAccount() : void
Tag
- name : string
- description : string
creates
Uses
to
Uses
creates/flags
Question
- id : int
- title : string
- content : string
- createdBy : User
- tags : Tag {list}
- followers : User {list}
- answers : Answer {list}
- comments : Comment {list}
- upvotes : int
- downvotes : int
- viewCount : int
- voteCount : int
- score : int
- creationDate : date/time
- modificationDate : date/time
- bounty : Bounty
- status : QuestionStatus
- closingReason : ClosingDetails
+ addComment(comment) : void
+ addBounty(bounty) : void
Comment
- id : int
- postedBy: User
- content : string
- flagCount : int
- voteCount : int
- upvotes : int
- creationDate : date/time
creates
creates
Notification
- notificationID : int
- createdOn : date/time
- content : string
+ sendNotification(account) : void
creates
<<Interface>>
Search
+ searchByTags(name) : Question {list}
+ searchByUsers(name) : Question {list}
+ searchByWords(words) : Question {list}
implements
Answer
- id : int
- content : string
- postedBy : User
- followers : User {list}
- comments : Comment {list}
- flagCount : int
- upvotes : int
- downvotes : int
- voteCount : int
- isAccepted : bool
- creationDate : date/time
+ addComment(comment) : void
Bounty
- reputationPoints : int
- expiryDate : date/time
+ updateReputationPoints() : void
SearchCatalog
- questionsUsingTags : Map<string, Tag {list}>
- questionsUsingUsers : Map<string, User {list}>
- questionsUsingWords : Map<string, string {list}>

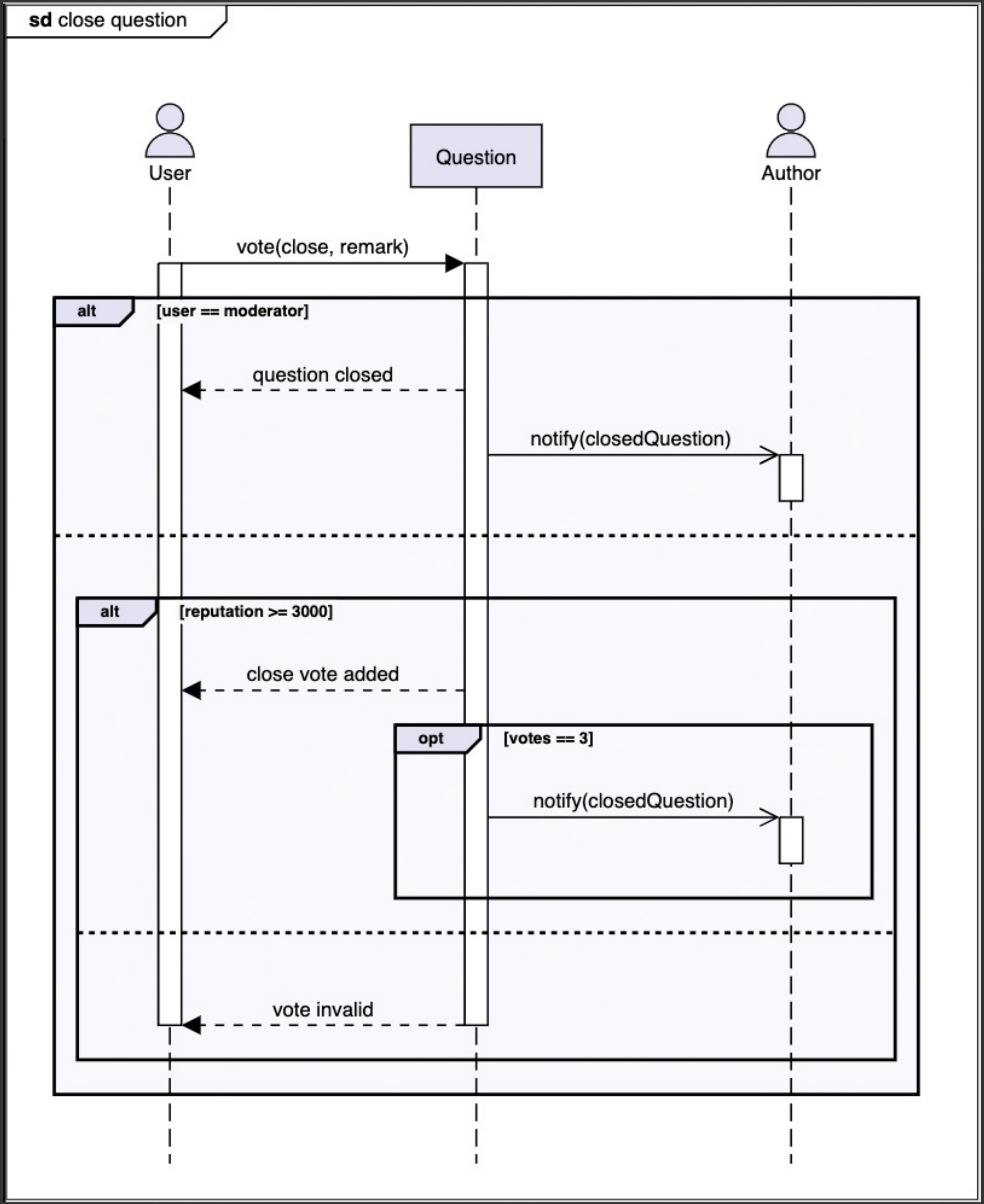
Addional Requirement

→ Save questions or answers

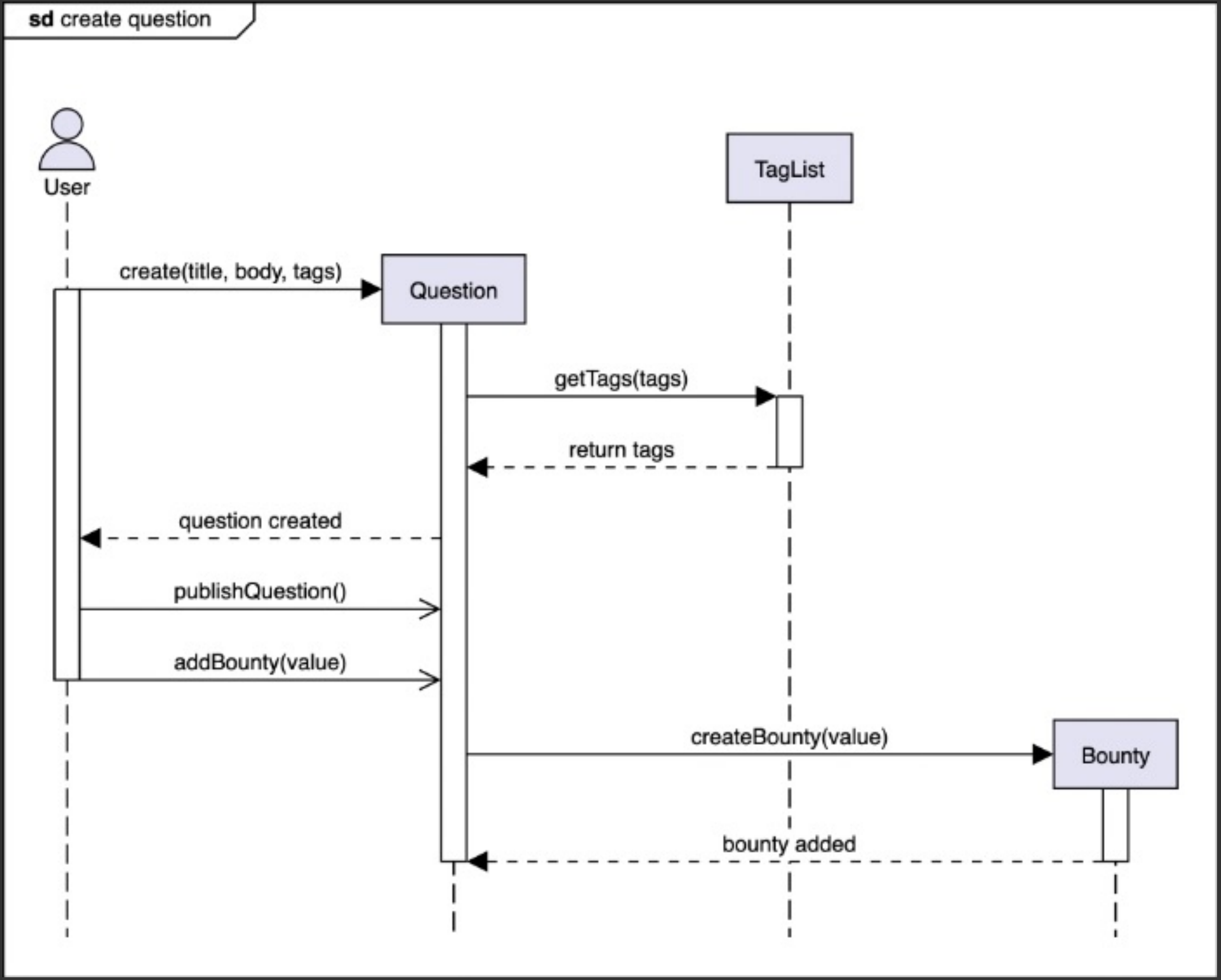


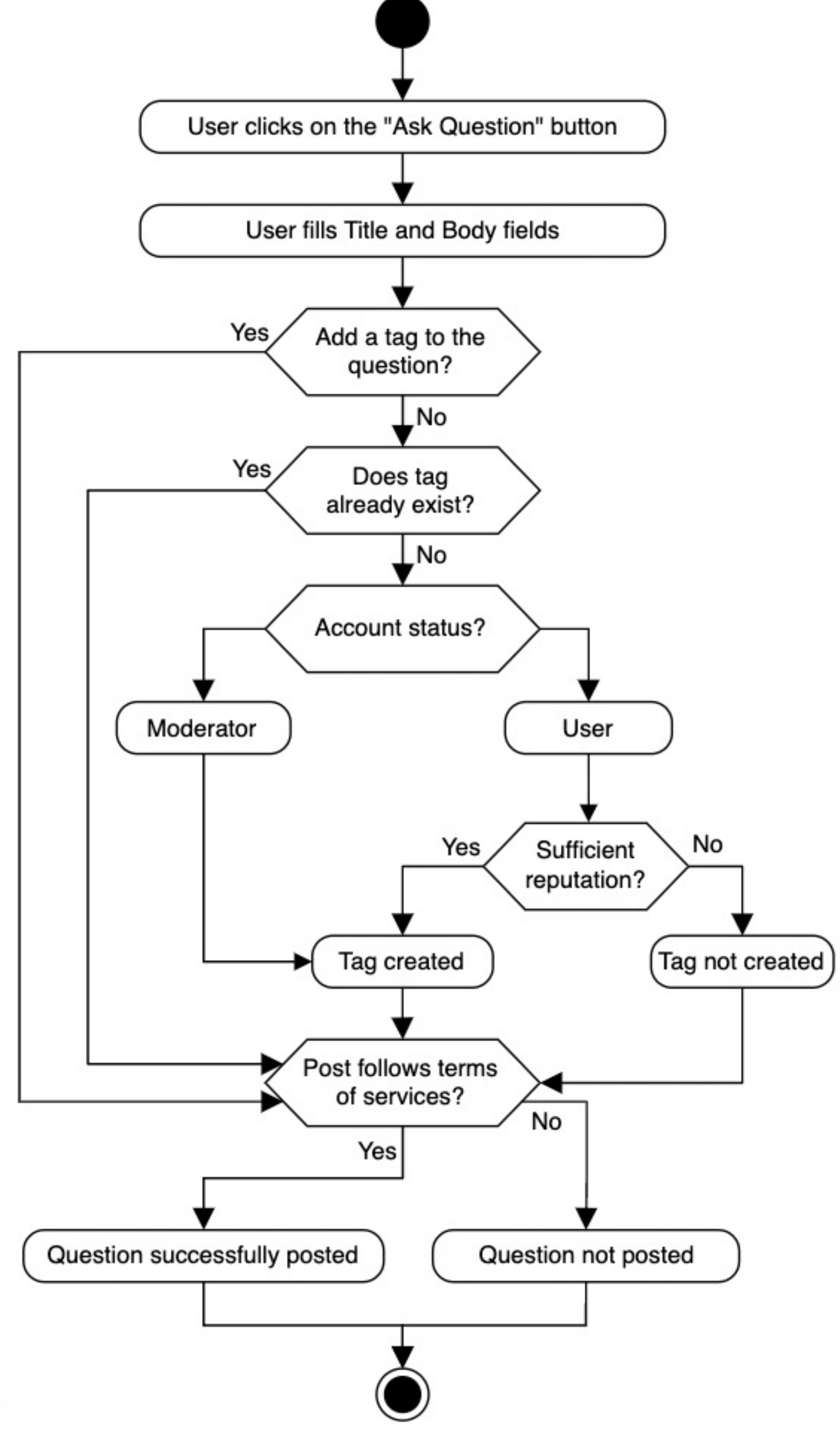
Question
- id : int
- title : string
- content : string
- createdBy : User
- tags : Tag {list}
- followers : User {list}
- answers : Answer {list}
- comments : Comment {list}
- upvotes : int
- downvotes : int
- viewCount : int
- voteCount : int
- score : int
- creationDate : date/time
- modificationDate : date/time
- bounty : Bounty
- status : QuestionStatus
- closingReason : ClosingDetails
+ addComment(comment) : void
+ addBounty(bounty) : void
Answer
- id : int
- content : string
- postedBy : User
- followers : User {list}
- comments : Comment {list}
- flagCount : int
- upvotes : int
- downvotes : int
- voteCount : int
- isAccepted : bool
- creationDate : date/time
+ addComment(comment) : void
User
- reputationPoints : int
- badges: Badges {list}
+ createQuestion() : bool
+ addAnswer() : bool
+ createComment() : bool
+ upvote(id) : void
+ downvote(id) : void
+ flagQuestion(question) : void
+ flagAnswer(answer) : void
+ voteToCloseQuestion(question) : void
+ voteToDeleteQuestion(question) : void
+ acceptAnswer(answer) : void
+ saveQuestion(question) : void
+ saveAnswer(answer) : void
Saves
- savedQuestions : Question {list}
- savedAnswers : Answers {list}

Sequence Diagram of Closing Question



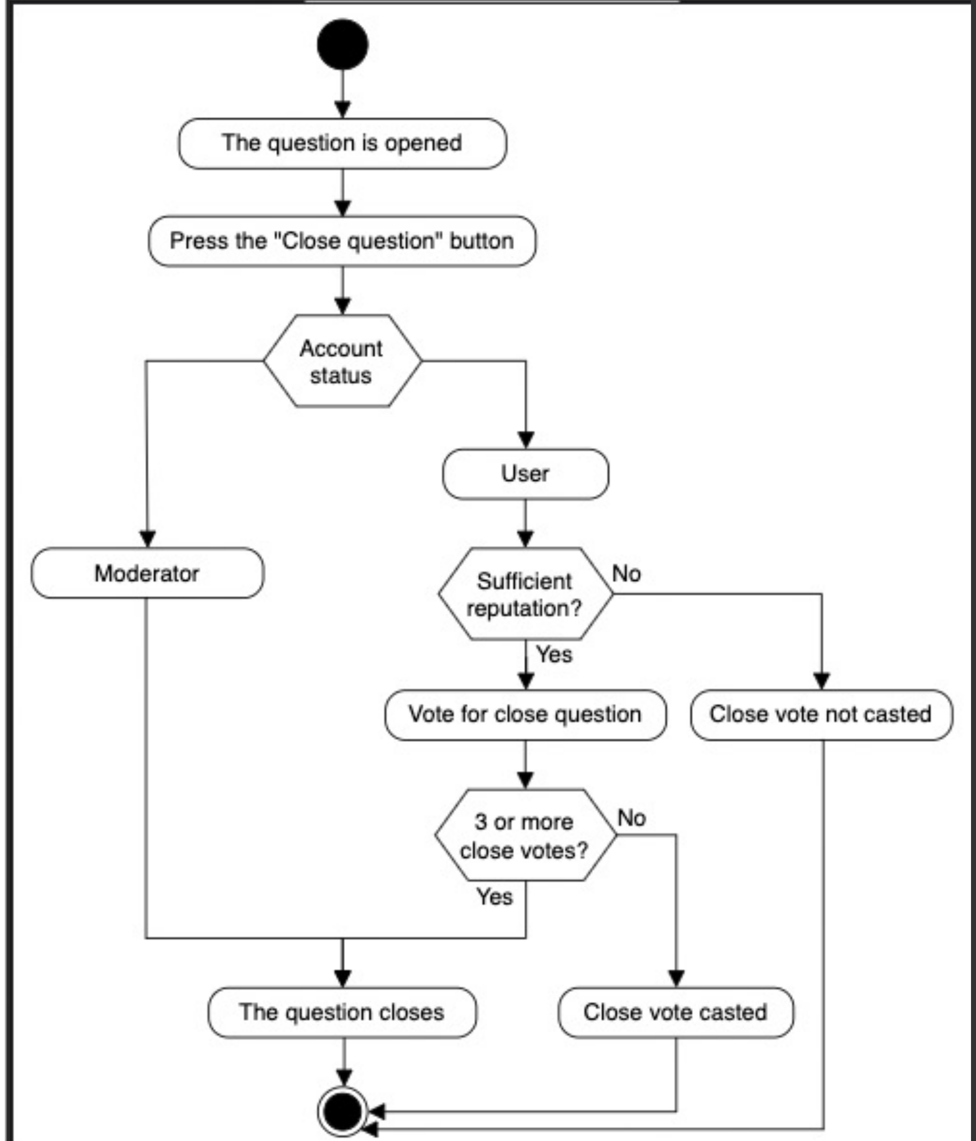
Sequence diagram of Create Question




User clicks on the "Ask Question" button
User fills Title and Body fields
Add a tag to the question?
Yes
No
Does tag already exist?
Yes
No
Account status?
Moderator
User
Sufficient reputation?
Yes
No
Tag created
Tag not created
Post follows terms of services?
Yes
No
Question successfully posted
Question not posted

→ sequence diagram to ask the questing

↑ sequence diagram to close the diagram



Code

1. Enumeration

```
enum AccountStatus {
  ACTIVE,
  BLOCKED,
  DISABLED
}

enum QuestionStatus {
  ACTIVE,
  CLOSED,
  FLAGGED,
  BOUNTIED
}

enum ClosingDetail {
  COMMUNITY_SPECIFIC_REASON,
  DUPLICATE,
  NEEDS_CLARITY,
  NEEDS_MORE_FOCUS,
  OPINION_BASED
}
```

2. Account

```
public class Account {
  private String accountId;
  private String username;
  private String password;
  private String name;
  private String email;
  private int phone;
  private AccountStatus
status;

  public boolean
resetPassword();
}
```

3. User, admin, moderator & guest

```
public class User {
  private int reputationPoints;
  private Account account;
  private List<Badge> badges;

  public boolean createQuestion(Question
question);
  public boolean addAnswer(Question, question,
Answer answer);
  public boolean createComment(Comment comment);
  public boolean createTag(Tag tag);
  public void flagQuestion(Question question);
  public void flagAnswer(Answer answer);
  public void upvote(int id);
  public void downvote(int id);
  public void voteToCloseQuestion(Question
question);
  public void voteToDeleteQuestion(Question
question);
  public void acceptAnswer(Answer answer);
}

public class Admin extends User {
  public boolean blockUser(User user);
  public boolean unblockUser(User user);
  public void awardBadge(User user, Badge badge);
}

public class Moderator extends User {
  public void closeQuestion(Question question);
  public void reopenQuestion(Question question);
  public void deleteQuestion(Question question);
  public void restoreQuestion(Question question);
  public void deleteAnswer(Answer answer);
}

public class Guest {
  public void registerAccount();
}
```

4. Question, Answer, Comment & bounty

```
public class Question {
  private int id;
  private String title;
  private String content;
  private User createdBy;
  private int upvotes;
  private int downvotes;
  private int viewCount;
  private int score;
  private int voteCount;
  private Date creationDate;
  private Date modificationDate;
  private QuestionStatus status;
  private ClosingDetails closingReason;
  private Bounty bounty;

  private List<Tag> tags;
  private List<Comment> comments;
  private List<Answer> answers;
  private List<User> followers;

  public void addComment(Comment comment);
  public void addBounty(Bounty bounty);
}

public class Comment {
  private int id;
  private String content;
  private int flagCount;
  private int upvotes;
  private Date creationDate;
  private User postedBy;
}

public class Answer {
  private int id;
  private String content;
  private int flagCount;
  private int voteCount;
  private int upvotes;
  private int downvotes;
  private boolean isAccepted;
  private Date creationTime;
  private User postedBy;

  private List<Comment> comments;
  private List<User> followers;

  public void addComment(Comment comment);
}

public class Bounty {
  private int reputationPoints;
  private Date expiryDate;
  public boolean updateReputationPoints(int reputation);
```

5. Badge, tag & taglist

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

public class Tag {
  private String name;
  private String description;
}

public class TagList {
  private HashMap<Tag, int> tagsCount;
  public void incrementTagCount();
  public void decrementTagCount();
}
```

6. Notification

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

  public boolean sendNotification(Account
account);
}
```

7 Search Catlog & Interface

```java
public interface Search {
  public List<Question> searchByTags(String name);
  public List<Question> searchByUsers(String name);
  public List<Question> searchByWords(String words);
}

public class SearchCatalog implements Search {
  private HashMap<String, List<Tag>> questionsUsingTags;
  private HashMap<String, List<User>> questionsUsingUsers;
  private HashMap<String, List<String>> questionsUsingWords;

  public List<Question> searchByTags(String name) {
    // functionality
  }

  public List<Question> searchByUsers(String name) {
    // functionality
  }

  public List<Question> searchByWords(String words) {
    // functionality
  }
}
```