User Documentation

Authors: Angela Kim, Veronica Pimenova, Crystal Cheng, Eric Lin

This is the user documentation for our project. It explains the features, setup instructions, and usage details.

Table of Contents

Feature Descriptions:

Feature 1: Anonymous Posting
The Anonymous Posting feature allows users to make a post without having their name/profile attached to the post publicly. This feature provides more flexibility for users, providing them with an option to post anonymously if they feel like hiding their identity.

Key Functionality:

- Mark post as anonymous: When drafting a post, users will see a checkbox that lets them modify the anonymous status of the post
- Post displays as anonymous: Upon posting, the post shows up with no profile picture or name associated with the post author

Feature 2: isAnswered
The `isAnswered` feature was implemented to track the status of topics in the forum and allow users to quickly identify which topics have received a satisfactory response. This feature is particularly useful in Q&A or support forums, where users may want to focus on unanswered queries. The `isAnswered` attribute is a Boolean flag attached to each topic, indicating whether the issue or question raised in the initial post has been adequately addressed.

Key Functionality:

- Mark Topics as Answered: Users with the necessary privileges can mark a topic as answered.
- Display Answer Status: The status (`Answered` or `Unanswered`) is displayed on the topic list view.
- Restrict Marking Privileges: The ability to toggle the `isAnswered` status can be restricted to specific users who posted.

Feature 3: Emoji Reactions
The Emoji Reactions feature allows users to express their sentiments on posts by reacting with emojis. This provides a lightweight and intuitive way for users to engage with posts without having to write a full response. Reactions can be added to any post within the forum, and the reactions are displayed directly below the post.

Key Functionality:

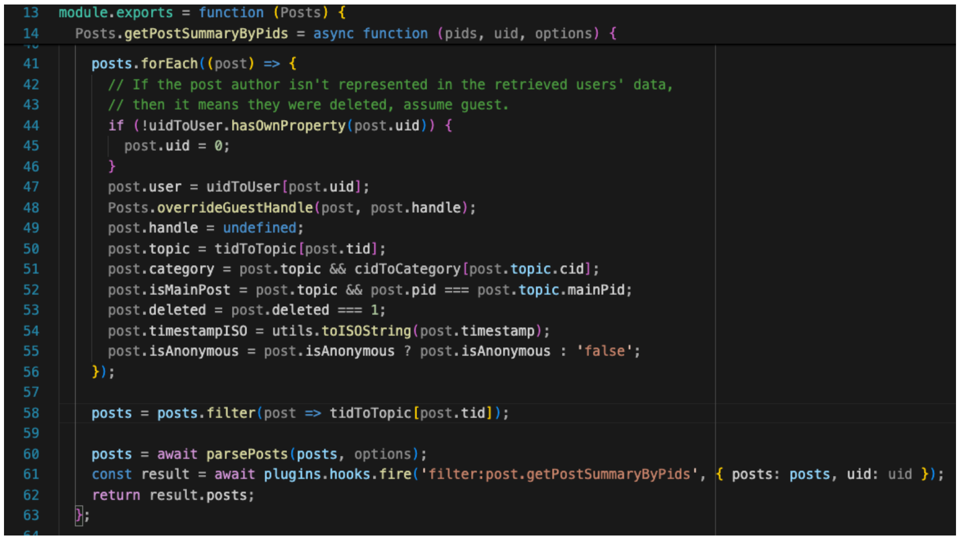
- React to Posts: Users can choose from a set of emojis to react to a post.
- View Reactions: The total number of reactions and the specific emojis used are displayed under each post, allowing users to see the sentiment of the community toward the post.
- Remove Reactions: Users can remove their reactions if they no longer wish to associate them with a post.

Implementation Details:

Feature 1:

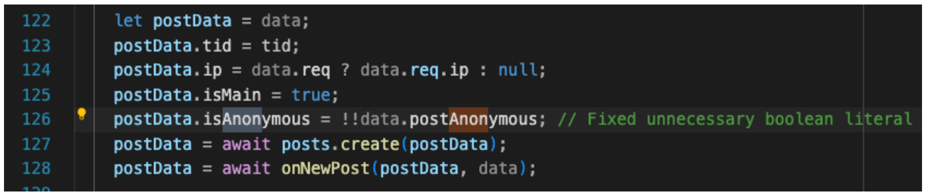
1. Backend Field handling: A field isAnonymous is added to handle the server requests associated with a post's anonymity

src/posts/summary.js

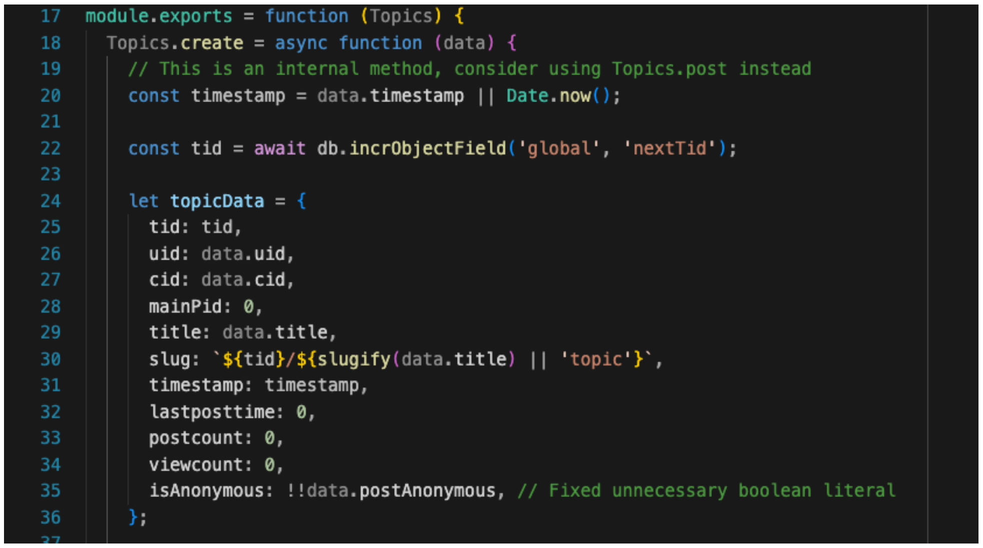


```javascript
module.exports = function (Posts) {
	Posts.getPostSummaryByPids = async function (pids, uid, options) {
		posts.forEach((post) => {
			// If the post author isn't represented in the retrieved users' data,
			// then it means they were deleted, assume guest.
			if (!uidToUser.hasOwnProperty(post.uid)) {
				post.uid = 0;
			}
			post.user = uidToUser[post.uid];
			Posts.overrideGuestHandle(post, post.handle);
			post.handle = undefined;
			post.topic = tidToTopic[post.tid];
			post.category = post.topic && cidToCategory[post.topic.cid];
			post.isMainPost = post.topic && post.pid === post.topic.mainPid;
			post.deleted = post.deleted === 1;
			post.timestampISO = utils.toISOString(post.timestamp);
			post.isAnonymous = post.isAnonymous ? post.isAnonymous : 'false';
		});

		posts = posts.filter(post => tidToTopic[post.tid]);

		posts = await parsePosts(posts, options);
		const result = await plugins.hooks.fire('filter:post.getPostSummaryByPids', { posts: posts, uid: uid });
		return result.posts;
	};
```

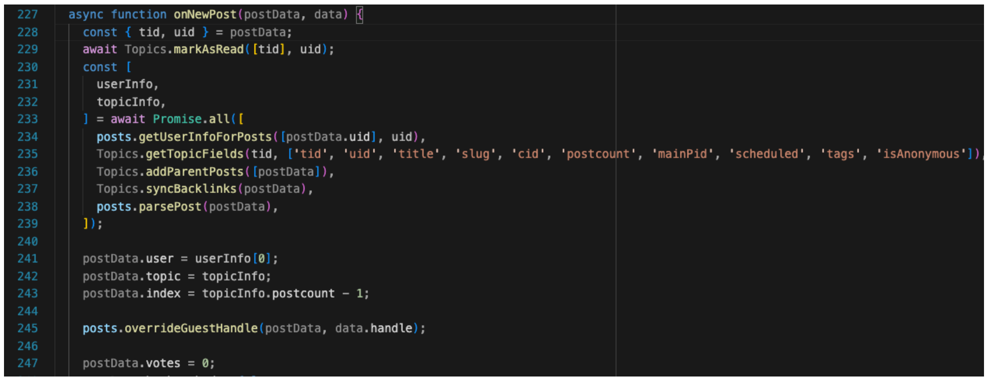
src/topics/create.js



```javascript
		let postData = data;
		postData.tid = tid;
		postData.ip = data.req ? data.req.ip : null;
		postData.isMain = true;
		postData.isAnonymous = !!data.postAnonymous; // Fixed unnecessary boolean literal
		postData = await posts.create(postData);
		postData = await onNewPost(postData, data);
```

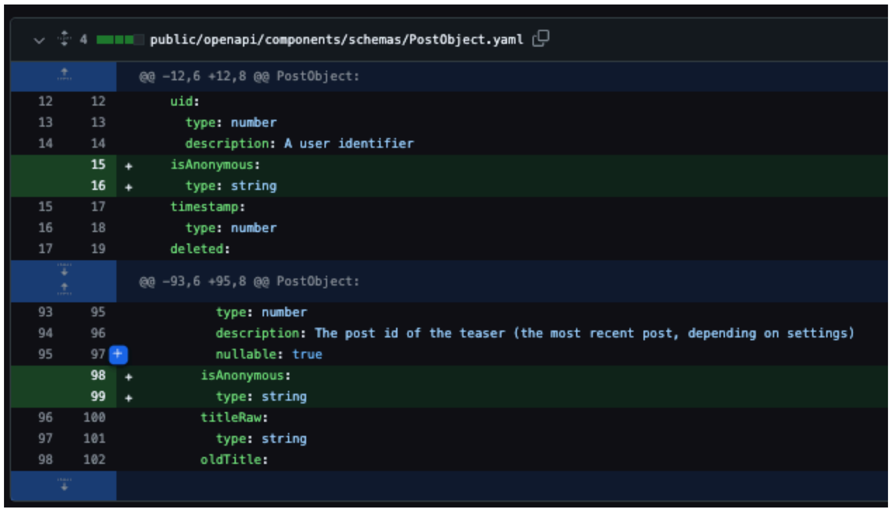


```javascript
module.exports = function (Topics) {
	Topics.create = async function (data) {
		// This is an internal method, consider using Topics.post instead
		const timestamp = data.timestamp || Date.now();

		const tid = await db.incrObjectField('global', 'nextTid');

		let topicData = {
			tid: tid,
			uid: data.uid,
			cid: data.cid,
			mainPid: 0,
			title: data.title,
			slug: `${tid}/${slugify(data.title) || 'topic'}`,
			timestamp: timestamp,
			lastposttime: 0,
			postcount: 0,
			viewcount: 0,
			isAnonymous: !!data.postAnonymous, // Fixed unnecessary boolean literal
		};
```

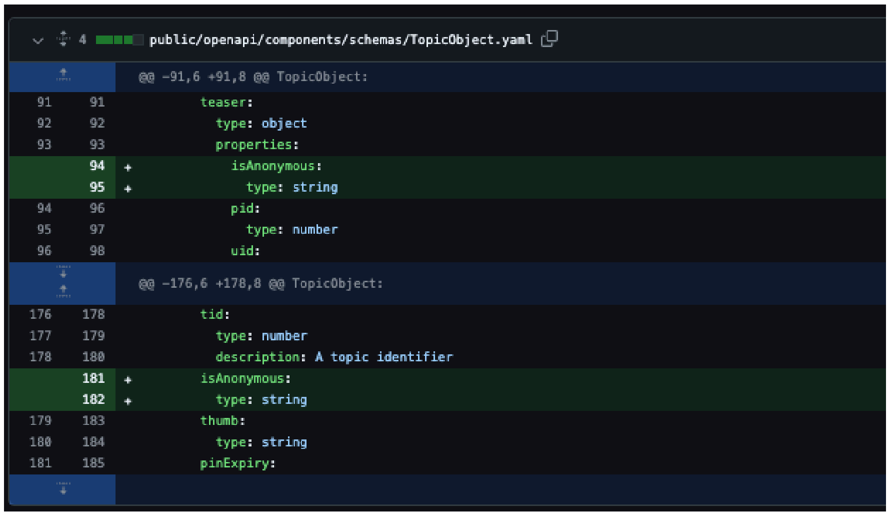


```javascript
async function onNewPost(postData, data) {
  const { tid, uid } = postData;
  await Topics.markAsRead([tid], uid);
  const [
    userInfo,
    topicInfo,
  ] = await Promise.all([
    posts.getUserInfoForPosts([postData.uid], uid),
    Topics.getTopicFields(tid, ['tid', 'uid', 'title', 'slug', 'cid', 'postcount', 'mainPid', 'scheduled', 'tags', 'isAnonymous'])
    Topics.addParentPosts([postData]),
    Topics.syncBacklinks(postData),
    posts.parsePost(postData),
  ]);

  postData.user = userInfo[0];
  postData.topic = topicInfo;
  postData.index = topicInfo.postcount - 1;

  posts.overrideGuestHandle(postData, data.handle);

  postData.votes = 0;
```

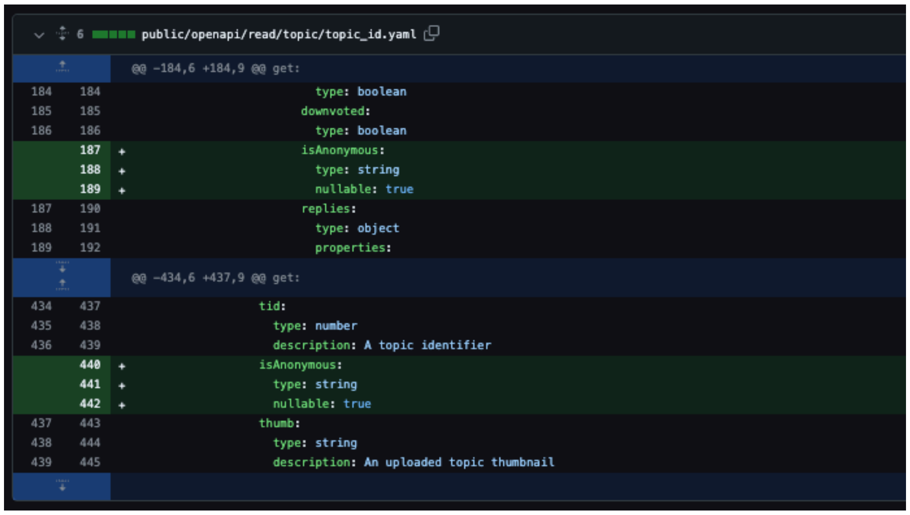
2. Backend Schema: The schema was expanded to include the “isAnonymous” field to match the response body of operations



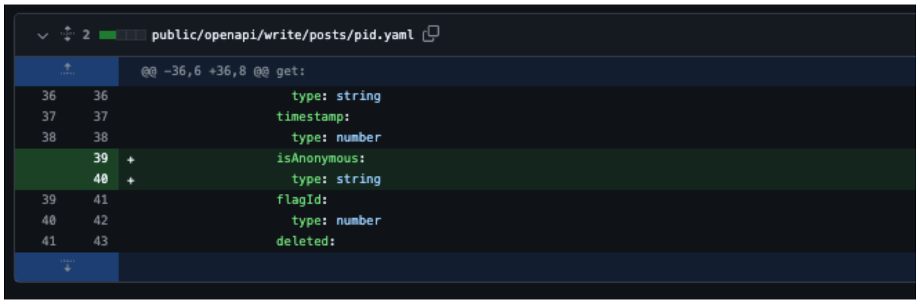
```diff
4 public/openapi/components/schemas/PostObject.yaml
@@ -12,6 +12,8 @@ PostObject:
      uid:
        type: number
        description: A user identifier
+     isAnonymous:
+       type: string
      timestamp:
        type: number
      deleted:
@@ -93,6 +95,8 @@ PostObject:
            type: number
            description: The post id of the teaser (the most recent post, depending on settings)
            nullable: true
+         isAnonymous:
+           type: string
          titleRaw:
            type: string
          oldTitle:
```



```diff
4 public/openapi/components/schemas/TopicObject.yaml
@@ -91,6 +91,8 @@ TopicObject:
          teaser:
            type: object
            properties:
+             isAnonymous:
+               type: string
              pid:
                type: number
              uid:
@@ -176,6 +178,8 @@ TopicObject:
          tid:
            type: number
            description: A topic identifier
+         isAnonymous:
+           type: string
          thumb:
            type: string
          pinExpiry:
```

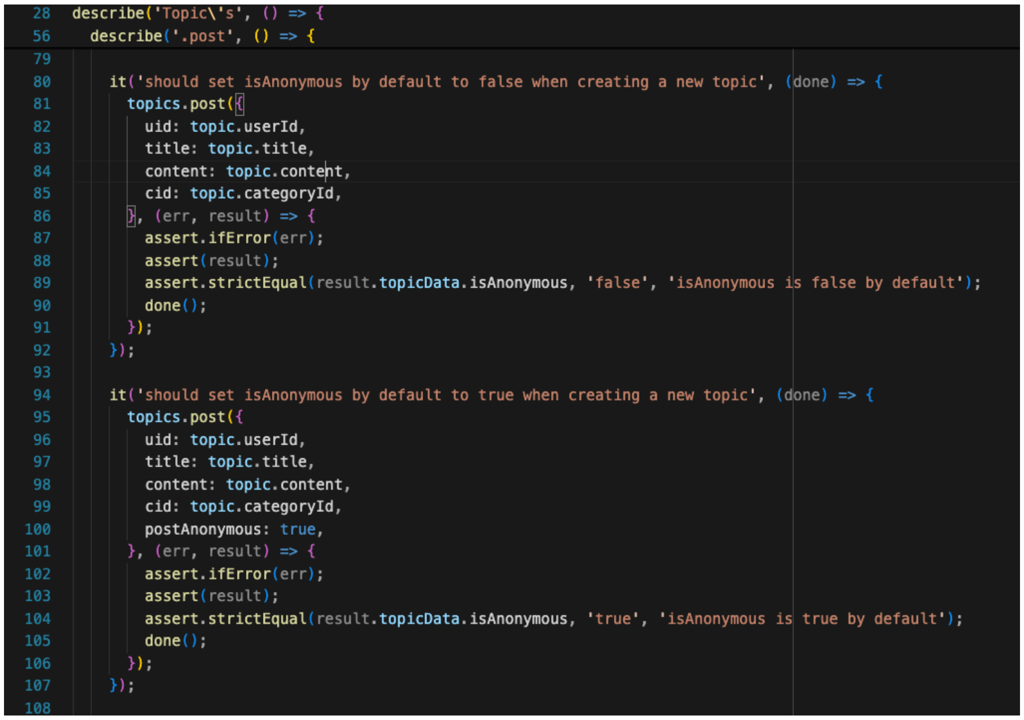


```diff
6 public/openapi/read/topic/topic_id.yaml
@@ -184,6 +184,9 @@ get:
                           type: boolean
                         downvoted:
                           type: boolean
+                        isAnonymous:
+                          type: string
+                          nullable: true
                         replies:
                           type: object
                           properties:
@@ -434,6 +437,9 @@ get:
                   tid:
                     type: number
                     description: A topic identifier
+                  isAnonymous:
+                    type: string
+                    nullable: true
                   thumb:
                     type: string
                     description: An uploaded topic thumbnail
```



```diff
2 public/openapi/write/posts/pid.yaml
@@ -36,6 +36,8 @@ get:
                       type: string
                     timestamp:
                       type: number
+                    isAnonymous:
+                      type: string
                     flagId:
                       type: number
                     deleted:
```

3. Testing for backend: Testing was added to the backend to ensure that our newly added field to verify the presence and ability to toggled through functions.

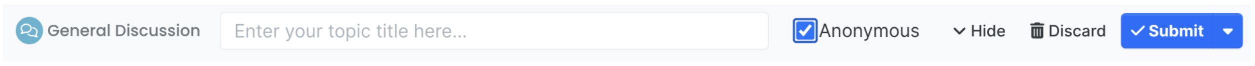


```javascript
describe('Topic\'s', () => {
  describe('.post', () => {

    it('should set isAnonymous by default to false when creating a new topic', (done) => {
      topics.post({
        uid: topic.userId,
        title: topic.title,
        content: topic.content,
        cid: topic.categoryId,
      }, (err, result) => {
        assert.ifError(err);
        assert(result);
        assert.strictEqual(result.topicData.isAnonymous, 'false', 'isAnonymous is false by default');
        done();
      });
    });

    it('should set isAnonymous by default to true when creating a new topic', (done) => {
      topics.post({
        uid: topic.userId,
        title: topic.title,
        content: topic.content,
        cid: topic.categoryId,
        postAnonymous: true,
      }, (err, result) => {
        assert.ifError(err);
        assert(result);
        assert.strictEqual(result.topicData.isAnonymous, 'true', 'isAnonymous is true by default');
        done();
      });
    });

```

4. Frontend for Feature 1

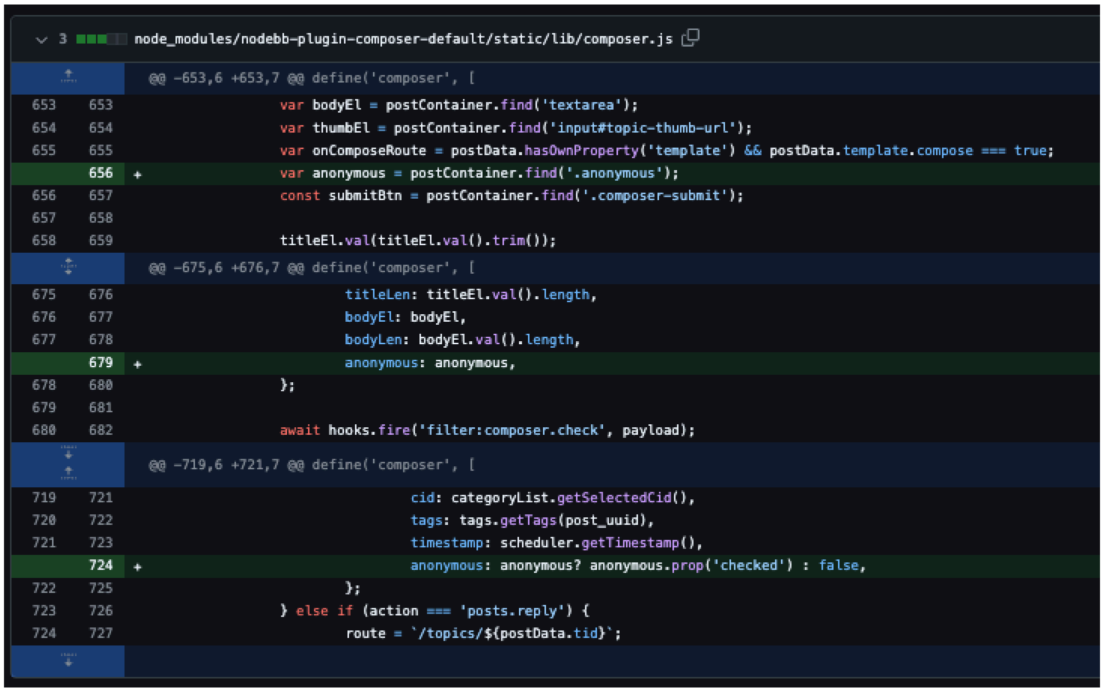
Frontend UI:

- **Checkbox Display**: A checkbox shows up right next to the topic title, handling the anonymous state of each post made



Integration:

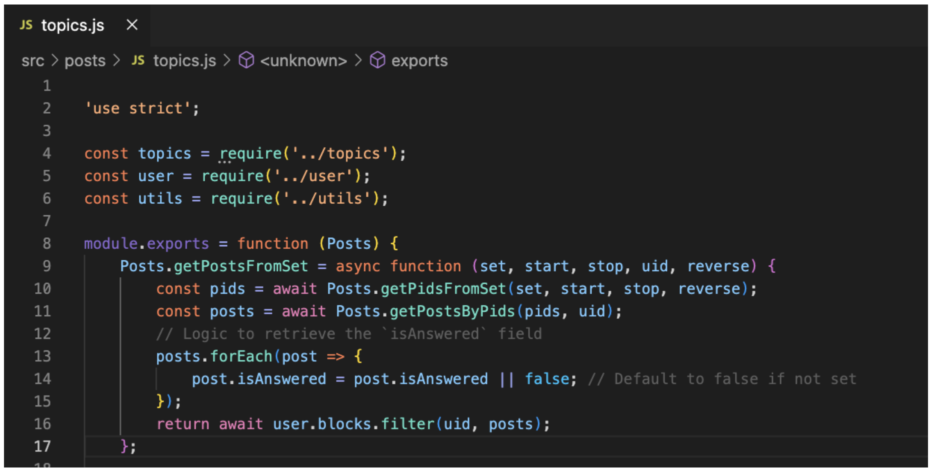
- When the checkbox is checked and a post is published, a frontend trigger reflects this change



```
3  node_modules/nodebb-plugin-composer-default/static/lib/composer.js

@@ -653,6 +653,7 @@ define('composer', [
                var bodyEl = postContainer.find('textarea');
                var thumbEl = postContainer.find('input#topic-thumb-url');
                var onComposeRoute = postData.hasOwnProperty('template') && postData.template.compose === true;
+               var anonymous = postContainer.find('.anonymous');
                const submitBtn = postContainer.find('.composer-submit');

                titleEl.val(titleEl.val().trim());
@@ -675,6 +676,7 @@ define('composer', [
                        titleLen: titleEl.val().length,
                        bodyEl: bodyEl,
                        bodyLen: bodyEl.val().length,
+                       anonymous: anonymous,
                };

                await hooks.fire('filter:composer.check', payload);
@@ -719,6 +721,7 @@ define('composer', [
                                cid: categoryList.getSelectedCid(),
                                tags: tags.getTags(post_uuid),
                                timestamp: scheduler.getTimestamp(),
+                               anonymous: anonymous? anonymous.prop('checked') : false,
                        };
                } else if (action === 'posts.reply') {
                        route = `/topics/${postData.tid}`;
```

Feature 2:

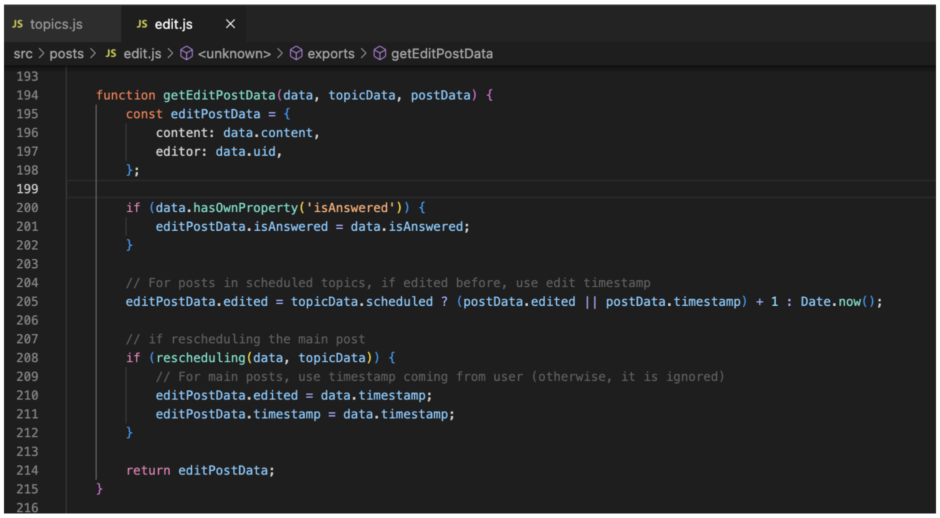
1. Backend Schema: A new column `isAnswered` was added to the `topics` table in the database to store the status of each topic.



topics.js

src > posts > topics.js > <unknown> > exports

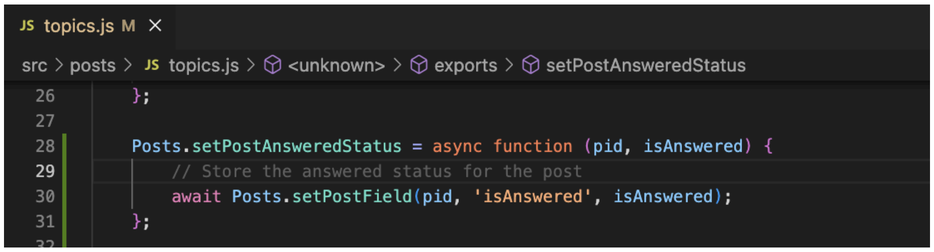
```javascript

'use strict';

const topics = require('../topics');
const user = require('../user');
const utils = require('../utils');

module.exports = function (Posts) {
    Posts.getPostsFromSet = async function (set, start, stop, uid, reverse) {
        const pids = await Posts.getPidsFromSet(set, start, stop, reverse);
        const posts = await Posts.getPostsByPids(pids, uid);
        // Logic to retrieve the `isAnswered` field
        posts.forEach(post => {
            post.isAnswered = post.isAnswered || false; // Default to false if not set
        });
        return await user.blocks.filter(uid, posts);
    };
```



topics.js edit.js

src > posts > edit.js > <unknown> > exports > getEditPostData

```javascript

    function getEditPostData(data, topicData, postData) {
        const editPostData = {
            content: data.content,
            editor: data.uid,
        };

        if (data.hasOwnProperty('isAnswered')) {
            editPostData.isAnswered = data.isAnswered;
        }

        // For posts in scheduled topics, if edited before, use edit timestamp
        editPostData.edited = topicData.scheduled ? (postData.edited || postData.timestamp) + 1 : Date.now();

        // if rescheduling the main post
        if (rescheduling(data, topicData)) {
            // For main posts, use timestamp coming from user (otherwise, it is ignored)
            editPostData.edited = data.timestamp;
            editPostData.timestamp = data.timestamp;
        }

        return editPostData;
    }

```

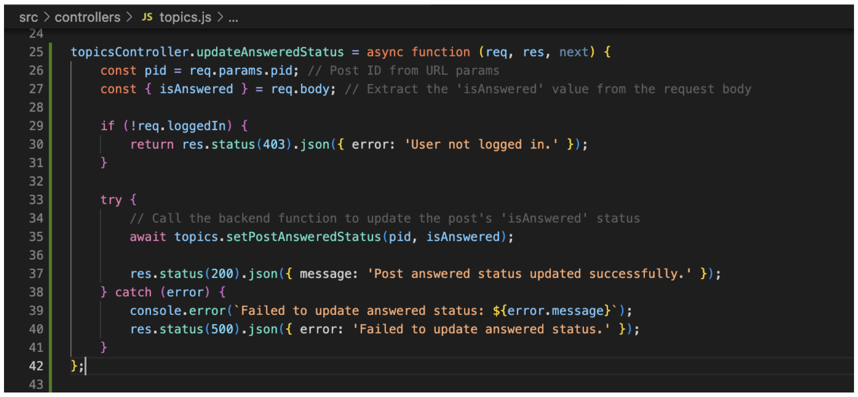


topics.js M

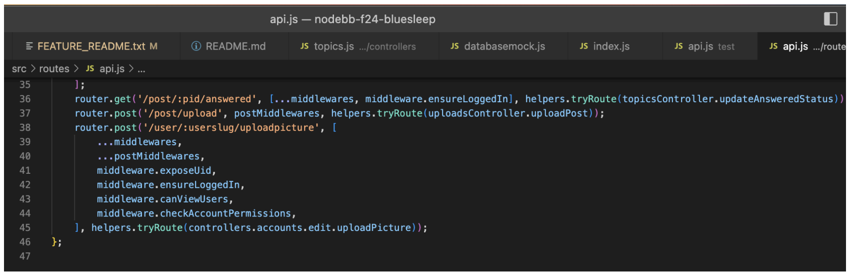
src > posts > topics.js > <unknown> > exports > setPostAnsweredStatus

```javascript
    };

    Posts.setPostAnsweredStatus = async function (pid, isAnswered) {
        // Store the answered status for the post
        await Posts.setPostField(pid, 'isAnswered', isAnswered);
    };
```

2. API Endpoint (Internal):
   - Endpoint: `/post/:pid/answered'
   - Method: `PUT`
   - Body: `{ "isAnswered": true }`
   - This API is used internally to toggle the `isAnswered` status.


src > controllers > JS topics.js > ...

```javascript
topicsController.updateAnsweredStatus = async function (req, res, next) {
    const pid = req.params.pid; // Post ID from URL params
    const { isAnswered } = req.body; // Extract the 'isAnswered' value from the request body

    if (!req.loggedIn) {
        return res.status(403).json({ error: 'User not logged in.' });
    }

    try {
        // Call the backend function to update the post's 'isAnswered' status
        await topics.setPostAnsweredStatus(pid, isAnswered);

        res.status(200).json({ message: 'Post answered status updated successfully.' });
    } catch (error) {
        console.error(`Failed to update answered status: ${error.message}`);
        res.status(500).json({ error: 'Failed to update answered status.' });
    }
};
```

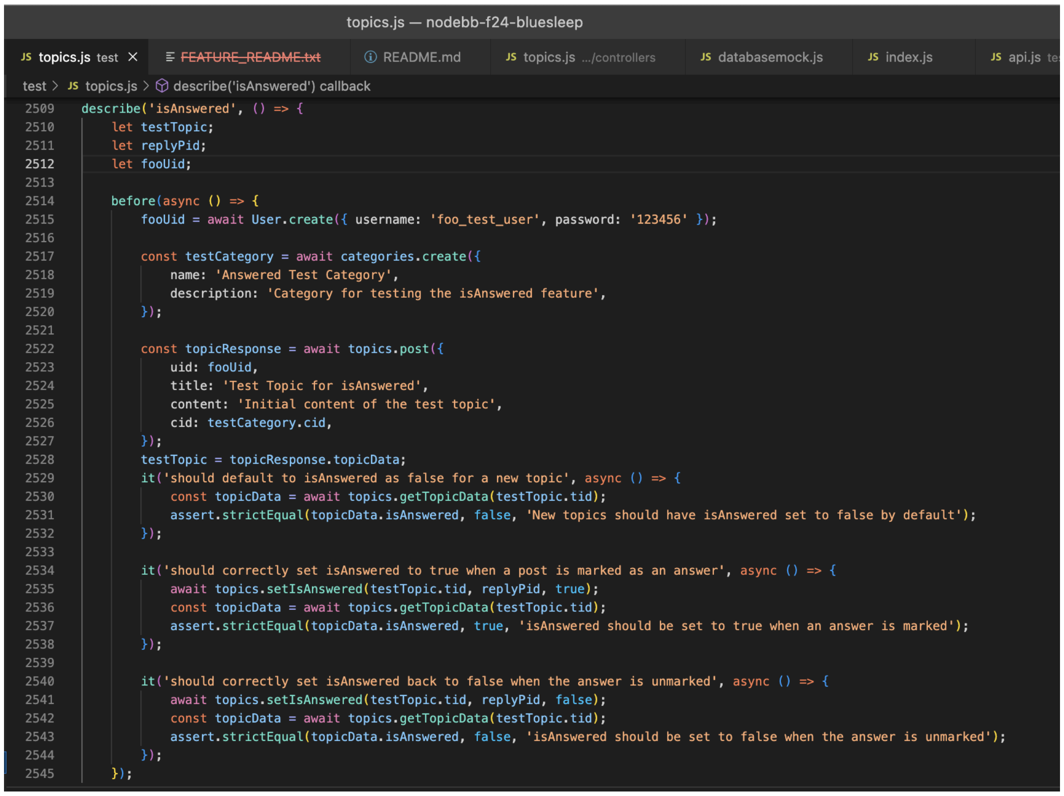

api.js — nodebb-f24-bluesleep

FEATURE_README.txt M | README.md | JS topics.js .../controllers | JS databasemock.js | JS index.js | JS api.js test | JS api.js .../route

src > routes > JS api.js > ...

```javascript
    ];
    router.get('/post/:pid/answered', [...middlewares, middleware.ensureLoggedIn], helpers.tryRoute(topicsController.updateAnsweredStatus))
    router.post('/post/upload', postMiddlewares, helpers.tryRoute(uploadsController.uploadPost));
    router.post('/user/:userslug/uploadpicture', [
        ...middlewares,
        ...postMiddlewares,
        middleware.exposeUid,
        middleware.ensureLoggedIn,
        middleware.canViewUsers,
        middleware.checkAccountPermissions,
    ], helpers.tryRoute(controllers.accounts.edit.uploadPicture));
};
```

3. Testing for Backend: Testing for backend has been added in order to ensure it runs as expected. These include whether the field exists, what is returned at each state, and whether it can be toggled using functions.


topics.js — nodebb-f24-bluesleep

JS topics.js test × | FEATURE_README.txt | README.md | JS topics.js .../controllers | JS databasemock.js | JS index.js | JS api.js te

test > JS topics.js > describe('isAnswered') callback

```javascript
describe('isAnswered', () => {
    let testTopic;
    let replyPid;
    let fooUid;

    before(async () => {
        fooUid = await User.create({ username: 'foo_test_user', password: '123456' });

        const testCategory = await categories.create({
            name: 'Answered Test Category',
            description: 'Category for testing the isAnswered feature',
        });

        const topicResponse = await topics.post({
            uid: fooUid,
            title: 'Test Topic for isAnswered',
            content: 'Initial content of the test topic',
            cid: testCategory.cid,
        });
        testTopic = topicResponse.topicData;
        it('should default to isAnswered as false for a new topic', async () => {
            const topicData = await topics.getTopicData(testTopic.tid);
            assert.strictEqual(topicData.isAnswered, false, 'New topics should have isAnswered set to false by default');
        });

        it('should correctly set isAnswered to true when a post is marked as an answer', async () => {
            await topics.setIsAnswered(testTopic.tid, replyPid, true);
            const topicData = await topics.getTopicData(testTopic.tid);
            assert.strictEqual(topicData.isAnswered, true, 'isAnswered should be set to true when an answer is marked');
        });

        it('should correctly set isAnswered back to false when the answer is unmarked', async () => {
            await topics.setIsAnswered(testTopic.tid, replyPid, false);
            const topicData = await topics.getTopicData(testTopic.tid);
            assert.strictEqual(topicData.isAnswered, false, 'isAnswered should be set to false when the answer is unmarked');
        });
    });
```

4. Frontend Integration: The status is displayed next to the topic title on the main topic list and within each topic's detailed view.
5. UI Changes: A checkbox or button labeled `Mark as Answered` is visible to users with sufficient permissions.

Frontend for feature 2:
Frontend UI:

- **Checkbox Display:** A checkbox is added to each post. The checkbox is checked if the post is marked as answered and unchecked if the post is unanswered.
- **UI Placement:** The checkbox appears next to the post title, both in the main topic list view and within the post detail view.

Integration:

- **Toggling Status:** When the checkbox is clicked, it triggers a frontend event that updates the post's status and visually toggles the checkbox to reflect the change.

Backend Communication:

- The frontend sends the updated status to the backend via an API call to ensure that the status is saved and updated in the database (the backend implementation is being handled separately).

Feature 3:
Backend Schema:
- No new column was added to the database. Emoji reactions leverage NodeBB's existing reactions system, which stores reactions associated with each post.

API Endpoint (Internal):
- Endpoint: /post/:pid/reaction
- Method: PUT
- Body: { "reaction": "😊" }
- This API endpoint is used internally to add a reaction to a post. The reaction field specifies which emoji to add.
- Endpoint: /post/:pid/reaction
- Method: DELETE
- Body: { "reaction": "😊" }
- This API endpoint is used internally to remove a reaction from a post.

Frontend Integration

Reacting to a Post:

- **Emoji Picker:** When viewing a post, users will see an emoji reaction icon next to the post content. Clicking on this icon opens the emoji picker, allowing users to select a reaction.

- **Displaying Reactions:** Once an emoji is selected, it appears below the post. The number of users who selected the same emoji is displayed next to each reaction.
- **Removing Reactions:** To remove a reaction, the user simply clicks on the emoji they selected, which will remove their reaction.

UI Changes:

- **Emoji Reaction Icon:** An emoji icon is placed next to each post, visible to all users.
- **Emoji Picker:** Clicking the emoji icon opens a dropdown or pop-up with a list of available emojis.
- **Reaction Display:** Selected emojis and their counts are displayed under the post content, providing a clear visual representation of the reactions.

Front-End Testing:
Ensure the checkbox appears next to each post's title and toggles between "answered" and "unanswered" states correctly.
Test that the checkbox state (checked or unchecked) is correctly reflected on the topic list view.
Verify that the checkbox can only be toggled by the original poster or users with sufficient permissions.

Back-End Testing:
Ensure the back-end API properly updates the post's isAnswered status in the database.
Test that the back-end correctly saves the new status when a user toggles the checkbox and retrieves the correct status when the page is loaded.

Edge Case Testing:
Test the functionality with multiple users to verify that only the user with proper permissions can toggle the status.
Ensure that the status change is immediately visible to all users once updated.

***Note: We decided to not focus too strongly on this feature (we may completely remove it in the coming weeks because it was much more difficult than we initially thought to implement, so the testing has not been fully implemented at this time as we focus on other features).* **

Usage:

Feature 1:

Setting a post as anonymous before posting:

1. Under a discussion, click Create Topic
2. Locate the anonymous checkbox near the textbox for drafting Post Title
3. Check that box, indicating that the post will be anonymous
4. Fill in some description, click Submit

Seeing the anonymous nature of the post:

- The post would display as a any regular post
- The only difference between this anonymous post and regular public posts is that the name and profile picture of the user is not visible.

Permissions:

- By default, only the original poster has permissions to adjust the anonymous nature of their post

Feature 2:

Marking a Post as Answered or Unanswered:

1. Navigate to the post you want to mark.
2. Locate the checkbox next to the post's title.
3. If the post is unanswered, click the checkbox to mark it as answered (the checkbox will be checked).
4. If the post is already marked as answered, you can uncheck the box to mark it as unanswered.

Viewing Answered/Unanswered Status:

- In the topic list view, you can see the answer status of all posts by looking at the checkbox next to each post.
- Within a specific post, the checkbox is also visible next to the title, indicating the current status.

Permissions:

- By default, only the original poster or users with sufficient permissions can toggle the answer status of a post.

Feature 3:
Adding a Reaction:

1. Navigate to the post you want to react to.
2. Click the emoji icon next to the post.

3. A list of emojis will appear.
4. Select the emoji you want to react with. It will be displayed under the post.

Removing a Reaction:

1. If you've already reacted to a post, your selected emoji will be highlighted under the post.
2. To remove your reaction, simply click on the emoji you selected. The reaction will be removed from the post.

Viewing Reactions:

- The emojis selected by users are shown under the post content.
- Next to each emoji is a count of how many users have selected that reaction.

Permissions:

- Any logged-in user can add or remove reactions from posts.
- There are no special permissions needed for adding or removing emoji reactions.