Overview

Replyke understands that one of the biggest challenges of managing an online community is moderation. Users can create posts and write comments that may not align with the values of your community. To address this, Replyke provides a robust moderation system.

When using the social comment section, a reporting functionality is already in place, requiring no additional configuration. Users can report comments that they believe violate the community's values, and these reports will appear in your project's dashboard for monitoring and appropriate action.

In addition to comment reporting, Replyke allows for entity reporting. Entities, like posts or other items, could also breach community guidelines. As developers have full control over the app's UI, implementing entity reporting requires minimal effort. Below, we'll walk through the process of setting up entity reporting in a React application.

Implementing Entity Reporting in React

The following example demonstrates how to implement entity reporting in a React application. This implementation ensures users can select a reason for their report and submit it for review.

```
import React, { useCallback, useMemo, useState } from "react";
import { useUser, useCreateReport, ReportReasonKey, reportReasons } from "@replyke/react-

const ReportPostModal = ({ reportedPost, onClose }) => {
  const { user } = useUser();
  const { createEntityReport } = useCreateReport();

  const [submitting, setSubmitting] = useState(false);
  const [reason, setReason] = useState<ReportReasonKey | null>(null);

  const buttonActive = useMemo(() => !!reason && !!reportedPost, [reason, reportedPost]);

  const handleSubmitReport = async () => {
    try {
      if (!reportedPost) throw new Error("No entity to report selected");
      if (!reason) throw new Error("No reason to report selected");
      if (!user) {
        alert("Please sign in or create an account to continue.");
```

```
          return;
        }

        setSubmitting(true);
        await createEntityReport({ targetId: reportedPost.id, reason });
        alert("Report submitted successfully.");
        onClose();
      } catch (err) {
        console.error("Failed to submit entity report")
      } finally {
        setSubmitting(false);
      }
    };

    return (
      <div>
        <h2>Submit a Report</h2>
        <p>Thank you for looking out for our community. Let us know what is happening, and
        <div className="report-reasons">
          {Object.entries(reportReasons).map(([key, value]) => (
            <button
              key={key}
              onClick={() => setReason(key as ReportReasonKey)}
              className={`reason-button ${key === reason ? "selected" : ""}`}
            >
              {value}
            </button>
          ))}
        </div>
        <button
          className="submit-button"
          onClick={handleSubmitReport}
          disabled={!buttonActive || submitting}
        >
          {submitting ? "Submitting..." : "Submit Report"}
        </button>
        <button onClick={onClose} className="close-button">Cancel</button>
      </div>
    );
  };

export default ReportPostModal;
```

Key Points

2. **Component Logic**: Users select a reason for their report from the provided options. Only logged-in users can submit reports.
3. **Styling**: Adapt the component's design to your app's styling requirements.

4. **Submission Feedback**: Provide feedback (e.g., alerts) for successful submissions or errors.

Dashboard Integration

Reports submitted by users will appear in your project's dashboard. From there, administrators can review reports and take appropriate actions, such as removing the reported entity or warning the user.

With Replyke's reporting system, maintaining the integrity of your community is straightforward and efficient.

Last updated on May 7, 2025