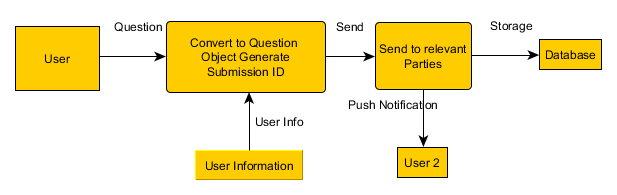
Users can generate two kinds of submissions: answer submissions, and question submissions.

**QuestionSubmission**

A question takes text and categories from the user and creates a **QuestionSubmission** object. The user enters the title for their and their questions body then presses submit. The system requests categories for the question such as course information through an alert. After the user confirms there categories a variety of information is passed to the backend of the system.

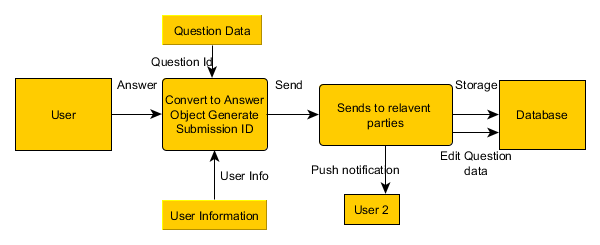
The system creates a **QuestionSubmission** object by setting a string equal to the title, setting **CategoryID,** an integer, to the selected category, initializing **QuestionAnswers**, an array, for **AnswerIDs** as an empty array, and generating the remainder of a **Submission** object which it inherits from. This involves setting a string body equal to the question text, setting an integer **AuthorID** equal to **UserID**, initializing **Score**, an integer, to zero and its **Status** to Active, setting **Date**, aJSdate object**,** equal to the current date and generating an **ID**, an integer, for itself. The information is then placed in the database and sent to other users based on the **CategoryID**. The following dataflow diagram illustrates this process:



**AnswerSubmission**

An answer takes text from the user and creates an answer object. The user enters their response to a given question and presses submit. The **ParentQuestionID**, **UserID** and answer text are then sent to the back end of the system. The backend takes this information and generates an **AnswerSubmission** object.

The answer object is created by setting **ParentQuestionID**, an integer, equal to the **AurthorID** of the question the user is answering and generating the remainder of a **Submission** object which **AnswerSubmission** objects inherit from. This involves setting **Body,** a string, equal to the answer text, setting **AuthorID**, an integer, equal to **UserID**, initializing **Score**, an integer, to zero and initializing **Status** to Active, setting **Date**, a JS Date object, equal to the current date, and generating an **ID**, an integer, for itself. The backend then updates the **QuestionSubmission** object in the database by placing its own **ID** in the array of answers found using the **ParentQuestionID**. Finally the backend pushes a notification to the user who asked the question by referencing the **AuthorId** on the **QuestionSubmission** object. The following dataflow diagram illustrates this process:



**AbuseReport**

A report takes a report type and a brief description of why they are reporting a submission to make an **AbuseReport**. The user selects either spam or language for the reason of the report. The backend then creates an **AbuseReport** object.

The backend generates **AbuseReportId**, **a**n integer, for the object, setting **Reason**, a string, to either spam or language, setting **Description**, a string to the description entered by the user, setting **SubmissionID,** an integer, to the reported objects submission id, setting its **Status** to active and setting **ReporterId**, an integer, to the reporters **UserID**. Once the report has been generated the **Status** of the flagged item is set to under review. The **AbuseReport** object is then place in storage to be reviewed by an admin later. The following dataflow diagram illustrates this process:

