## Submission

**Id**

All submissions have an integer ID generated when they are created in order to identify them. For a submission this is called its ID.

**AuthorId**

The AuthorID is the a user id associated with the user

**Body**

All submissions have a body associated with them. For an answer this will be the entirety of the text portion. A question also contains a title but a submission will contain the bulk of a question.

**Status**

The status of a submission is one of five states. It defaults to active on create and moves to archived after it has expired. If it is reported it is changed to in review where it is checked by an admin and changed either back to active, changed to deleted if it’s inappropriate or closed if it’s simply a poor question.

**Score**

This is the score that is associated with the submission. It is an integer value and changed as the students go.

**Date**

This is a JS date object and used to identify the time a submission was generated.

## User

Comprised of:

**UserId**

This is a unique integer value assigned when a user first signs up for the application using a myLaurier email address and is used to identify specific

**QuestionsAsked**

This is a list of submission ids associated with questions asked by a given User account

**AnswersProvided**

This is a list of submission ids associated with answers given through a User account

**State**

This is the state associated with a given user account and consists of the default of active, suspended which occurs as a discipline for an infraction, and deleted which is used when a user must be removed from the system.

**Score**

This is an integer associated with a user account that keeps track of the total score of their questions and answers.

## Notification

**Id**

This is a unique integer value created for a submission that a notification uses to identify what it is gathering from.

**Type**

Type specifies what kind of notification it is. This will specify if the notification is an answer or a question so the system can determine where those are supposed to go.

**Description**

This is a portion of the text within a submission object that relates to this given.

**Title**

If the notification is a question this contains the title of that question. If it is an answer it contains the title of the question that the answer is under.

**Date**

A JS date object used to show when a notification was generated.

**UserId**

A user ID associated with the User who generated the submission that the question goes with.

**Status**

This is the state associated with a notification object and consists of the default of unread and after a user check there notification it is changed to read.

## ProfileQuestionEntry

**Id**

This is an ID used by a given question

**Title**

This is the question that the ProfileQuestionEntry asks and is stored as a string.

**Description**

This is the possible ways to answer the question in the ProfileQuestionEntry  
**Value**

This is a string containing the answer they chose from the description.

AuthenticationToken

**Id**

The unique ID associated with an AuthenticationToken

**OwnerId**

This is the ID of the user who caused the AuthenticationToken to be spawned.

**Hash**

This is a crypto-secure hashed string of a secret key provided by an **OAuth** provider

**ExpiryDate**

A JS date object stating when the token will expire this is a fixed time after the issueDate

**IssueDate**

A JS date object stating when the token was created.

## AuthenticationController

AttemptAuthenticationForUser(username :string, key :String): Boolean

Takes in a Username and a string and returns whether or not the username has a valid hash key associated with it and is used to log in.

AuthenticationController()

this is the constructor for the authentication controller

ExpireSession()

This expires an authentication key

GetOAuthSessionKey(): String

This gets a session key from the OAuth provider and returns it as a string

ValidateToken(authenticationToken: AuthenticationToken): Boolean

This tells you whether a token is valid.

## NotificationController

FilterReadNotifications() : List <Notification>

This looks at all the notifications the user has cached and read and returns a list of them.

FilterUnreadNotifications(): List<Notification>

This looks at all the notifications the user has cached and not read and returns a list of them.

NotificationController()

This is a constructor

ProcessNotification(notification: Notification)

This is used to process a notification and decide what users to send it to.

SendNotification(notification: Notification)

This is used to send notifications to specific users.

## ServerController

FetchNotifications(): List<Notification>

Gathers notifications for use by the server.

FetchSubmissionFrom ServerById(id:Integer): QuestionSubmisison

Grabs a specific notification from the server using a question id to locate it

FetchUserFromServerById(id: Integer): User

Fetches a user account from the server by using the users id

ServerController()

A constructor for the server controller.

UpdateSubmissionOnServer(submission: Submission)

Takes a submission Object and uses its id to find and then overwrite the Submission data that was on the server with the Submission Object passed in to this function.

UpdateUserOnServer(user:User)

Takes a User Object and uses its id to find and then overwrite the User Object that was on the server with the User Object passed in to this function.

## QuestionController

FetchQuestionSubmission(): QuestionSubmission

ProcessQuestionSubmission(submission: QuestionSubmission)

QuestionController()

UpdateQuestionSubmission(submission :QuestionSubmission)

## AnswerControl

AnswerController()

FetchAnswerSubmission() : answer submission

This fetches an answer

ProcessAnswerSubmission(submission :AnswerSubmission)

UpdateAnswerSubmission(submission: AnswerSubmission)