Revisions for the revised design document are in blue

Overview

Many MIT students find it helpful to be able to work with others while doing psets or studying for tests; however, it can be difficult for students to find others to work with. To address this issue, we designed Cloop, a web application designed to allow MIT students to find classmates, build study groups, communicate with other students, and share resources in a comfortable, student-oriented forum. It features pages for each class offered at MIT, populated with all Cloop-registered users who are currently in each class. Users may make posts or comments on any of the class pages that they are associated with.

Purposes:

1) Finding classmates: Many students do not know how to find or reach out to other students in their classes.
2) Build study groups: Even if students find others in their classes, it may still be difficult for them to reach out for collaboration.
3) Communicate with other students: Students who may not want to form study groups or meet in person may still have questions or points of discussion for their peers.
4) Share resources: Students find it helpful to share resources such as notes, links, or other media.

On initial glance, Piazza and Facebook (or other social media platforms) have some aspects of what we are trying to achieve. However, neither of them provide a satisfactory solution to our problems. Piazza does not provide a comfortable place for students to interact because it is moderated by instructors and TAs; furthermore, many students feel scared to share their opinions out loud and instead post anonymously. While Facebook provides groups for general topics and purposes, it is not specifically structured for classes and education; users of Facebook are also limited by who they already know, and cannot reach out to those who are not already their friends.

Concepts

Class

- Purpose: To create an environment where MIT registered users in the same classes can interact freely
- Operational Principle: If a student joins a class, then they are added to the class page and can post or comment on the page, which they can navigate to via tabs on the user's interface. Students can also see other students' posts and comments in the class page. This allows MIT students in the same class to interact with one another via posts and comments on the same page.

Post

- Purpose: To allow MIT registered users to share their ideas and ask questions.
- Operational Principle: If a student creates a post on a class page, then their post will be displayed for everyone in the class to see, upvote, flag, or comment on. Students can have posts with text or resources - allowing them to ask questions or share class notes in the form of pdf or picture files.

Resource

- Purpose: To allow MIT registered users to share resources such as notes, powerpoints, etc.
- Operational Principle: If a student clicks the upload resource button on a class page, then they can choose a file to display for everyone in the class to see, upvote, flag, or comment on. Students are able to upload a variety of resources with different common file formats.

Comment

- Purpose: To allow MIT registered users to reply or give feedback to a post.
- Operational Principle: Each post has a comment button on it that users can click. If a user clicks the comment button, then they can write a text response to the post that is viewable by everyone in the class group. This allows students to reply and give feedback to certain posts. Students are also allowed to comment on their own post if they want to include more information for their post.

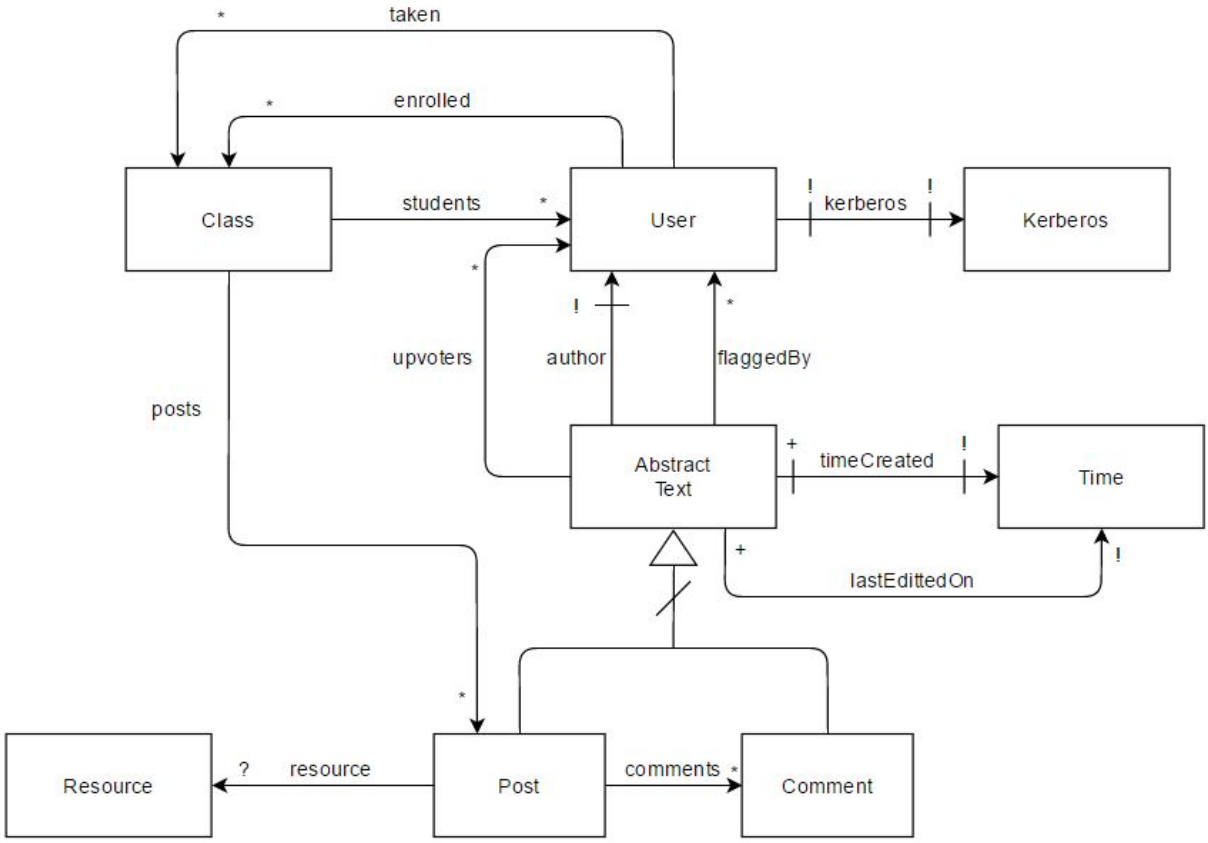
Flag

- Purpose: To allow users to secretly flag bad content for removal
- Operational Principle: Each post or comment has a ‘flag’ button that users can click. If enough different people flag it, it is automatically removed. If a user has enough removed comments, then they will receive a warning or penalty. This will allow bad posts/comments to be removed from the group in order to maintain the purpose of the class page.

Upvote

- Purpose: To allow MIT registered users to express that they like a post or comment
- Operational Principle: If a student upvotes a post or comment, then that post or comment will gain one upvote. Posts will display their number of upvotes. Posts and comments can be sorted by number of upvotes. This will allow quality content to be seen by users first.

Data Model:



Textual Constraints:

A User in the *students* of a Class has that Class in its *enrolled* relation

A User can only appear at most once in the *upvoters* of a text element (Post/Comment)

The mutable *lastEdittedOn* relation can only be modified to a more recent time (not a previous time)

No two Users have the same *kerberos* (email)

Model Explanations:

A post can be flagged by different users. After a set number of flags, the post will be removed from a class's *posts*.

The *lastEdittedOn* relation is from Text to the most recent time that the Text object was edited.

A Resource represents a file that may attachd to a post, so as to upload a "resource" with the post. It is its own concept, but its operational principle requires a Post for submission.

Model Insights:

In order to keep track of the most recent time edited in *lastEdittedOn,* the relation has to be mutable, as every time a Post/Comment is edited, the relation changes value.

If the relation was kept as *edittedOn*, the relation name suggests it should be immutable in the sense that one cannot change when an edit was made, and that said relation should contain multiple times, one for each edit.

The Kerberos is equivalent to email for MIT students - insight as to representation & implementation.
A class only exists if it has a student. While in the design and conceptual sense, *students* points to zero or more students, in the practical sense, *students* will have one or more students.

Threat Model:

- Potential attackers (as users) should only have access to the login and register page if they are not logged in.
- Potential attackers might want to steal other users' identities in order to gain access to the class pages and user pages in the web app. This can be done through stolen credentials or a false IP address.
- Potential attackers might try to modify data within the app itself or as it flows over a network.
- Potential attackers might try to deny service and make the application unavailable, either by passing input data that can crash the process or by overloading a server with requests to consume available system resources.

Security Concerns:

- To make sure users actually attend MIT, we will require users to register with an MIT email (formerly MIT certificates). Then, we will send a confirmation email to the email address to make sure that the user actually owns the @mit.edu address.
- To protect intellectual property and leaking of MIT exclusive material (ie. through posting resources), we require users to log in.
- To protect against injection attacks, XSS, and DoS, we will use templating (ie. Handlebars) and try to implement other strict input validations for fields and special characters
- To protect against CSRF, we will attempt to implement hidden tokens for each session or auto logout the user after a period of inactivity.
- To protect against spoofing user identity and tampering with data, we will not store secrets in plaintext.

Challenges:

Risks:

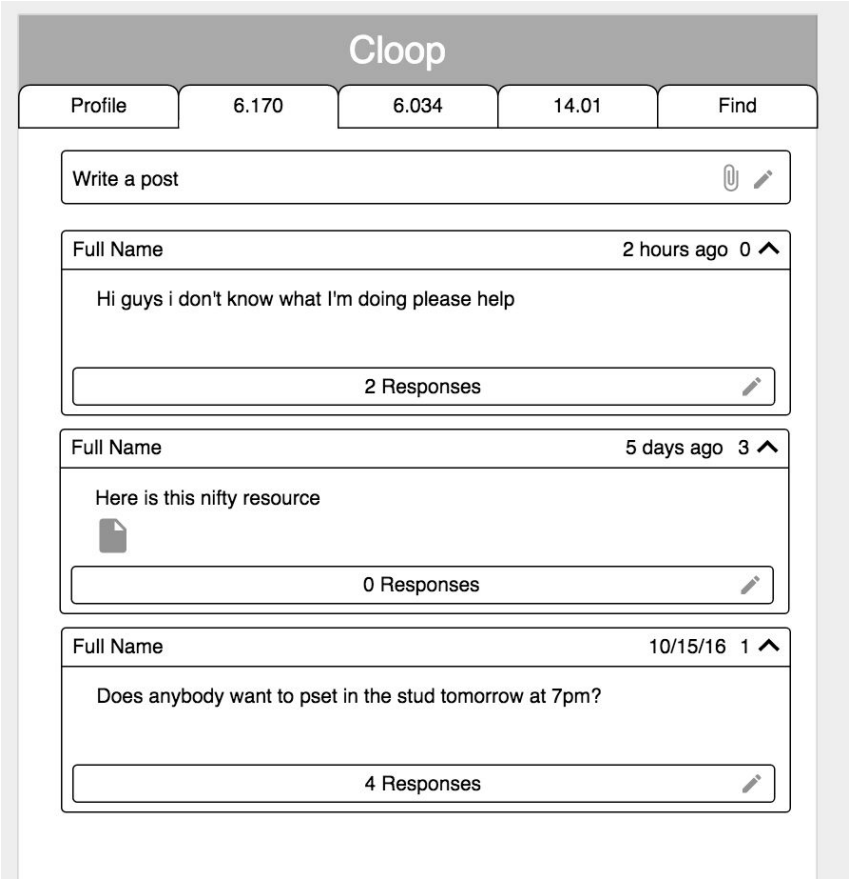
- Students may enroll in classes that they are not actually taking and disrupt the class page
    - Solutions: Posts or comments may be flagged, and students with too many removed posts or comments will be warned or penalized
- Students may abuse the flagging system and try to get every post or comment deleted
    - Solutions: Students have a limit on the number of flags (10 flags) they can give out every 12 hours

- Not enough students may use the app and it may die very quickly
    - Solutions: We will try our hardest to make this app the best that we can, so that users will want to keep using it.
- Students may spam class pages
    - Solutions: Posts or comments may be flagged, and students with too many removed posts or comments will be warned or penalized

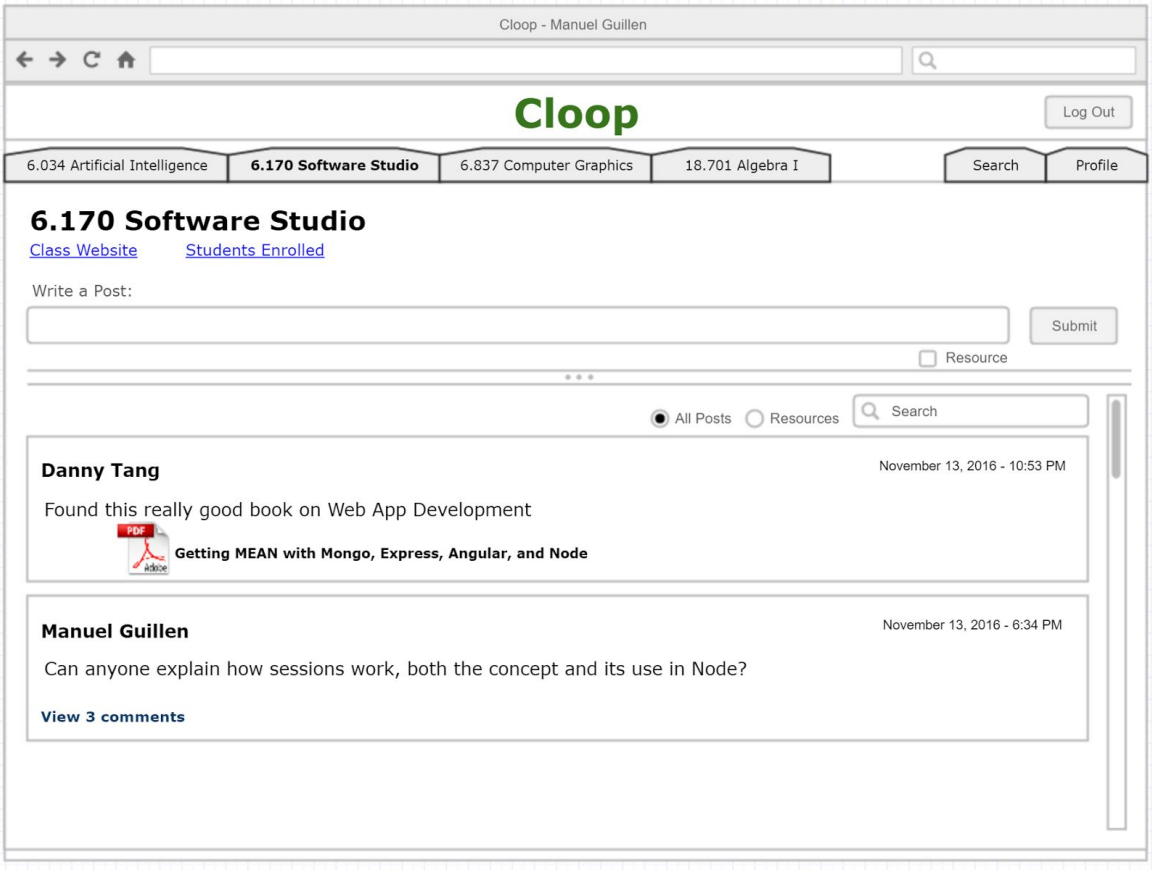
Design choices:

- We allow users to flag “bad” posts instead of downvoting them, because we don’t want to discourage or scare users from posting if they think they will only receive downvotes
- We allowed users to post resources because other students might find it helpful to look at notes or links. We chose to allow the posting of resources in posts to make it easier for students to access everything on one page (as opposed to a question/answer page and a resources page).
- We set the homepage as one of the user’s classes instead of the user’s profile (if user has not classes added, then the homepage will be their profile + an add class option). This caters to the user in the sense that when the user visits the web app, they want to see their class’ feed instead of their own profile.
- Instead of wiping a class’s page after every semester, we decided to create separate pages for every offering of the class. Old class pages will be archived. We did this so that old information will not be lost, in case anyone wants to refer back to it.
- We decided to disallow TA/instructor moderators - all users will be on the same platform with the same privileges - to reduce dependency on authority answers (e.g. disregarding student posts and waiting for an instructor answer), and to allow cloop to be more of a free discussion forum rather than a strict question answer format.
- We do not allow for the option to post anonymously because we want the Cloop community to be more open and comfortable. We don’t want the same environment as in Piazza, where students are afraid to post under their own name because they don’t want to be judged.
- We have decided not to allow student searching (only classes) to preserve student privacy (having every user’s class schedule available upon search might lead to undesirable situations)
- Users cannot create classes. This is to ensure that all classes have a specific name, and there are no classes with different names that are actually the same class.
- We have decided to narrow the types of resources we will support pdfs, to make resources a more standardized format (instead of dealing with .txt, .png, etc) and display through a node package (cloud storage). If we have time, we will also extend our app to support different types of files and extensions.
- Archived classes do not allow users to post/comment. Past users who have taken the class can see the archived class page. Users who were not in the archived class while it was still an active class are not allowed to join the archived class (archived classes will not be searchable). We chose this to ensure that users do not take advantage of resources from previous years.

UI Design:
The design of the user interface consists of a single main view, for which almost all of the relevant data is displayed. From this view, information about a registered class shows up on a class page, along with the collection of posts for that class. The registered class shows up like a tab on the page (design choice) - one for each registered class. Furthermore, the content of the page is determined by the opened class tab. The UI then allows on the open class to add posts and skim through the collection of posts for a class. The wireframe for this main page looks like so:

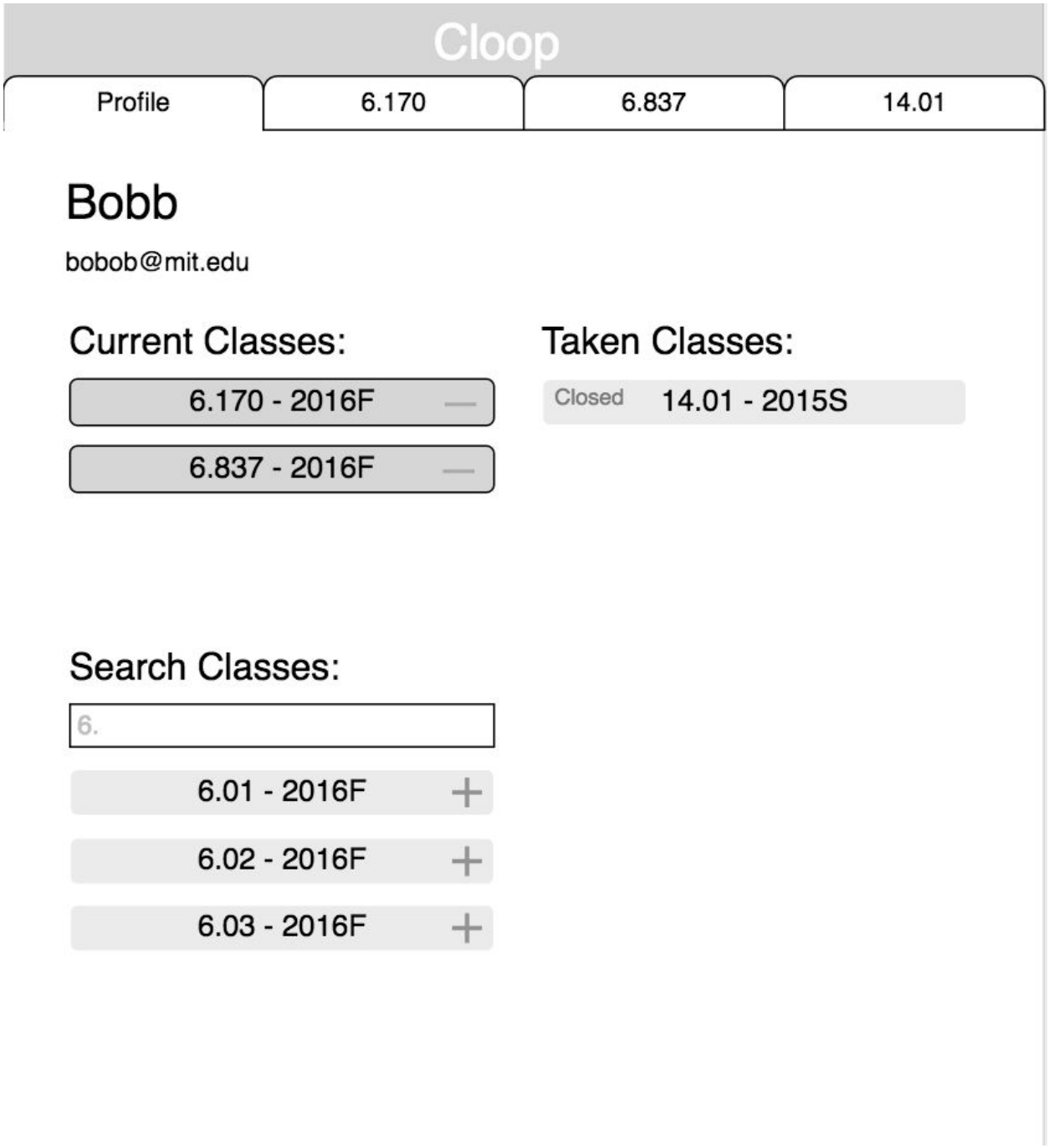


An example of a more thorough design implementation that could arise from this general UI specification is as follows:



Both wireframes capture the important aspects of the UI design, and have details that will end up being implemented in the interface. Following the implementation of the main view as described, we also have a login page - nothing special, just a log in form. Also, a new page is viewed when accessing one's profile, allowing for registration of more or less classes:

We also plan to implement a user page for easy class management. This page will include the user's information (name, email), active and past (closed/archived) classes, and a search bar for adding new classes.



From here, other users and other classes can be found, and their information (not the entire collection of posts, but characteristic data of the class) is displayed in a small popup on the middle of the screen. The choice of popup rather than new page was chosen for the UI as a consequence of the relatively small amount of data needed to be displayed for a user profile or a class profile, and for ease of use.

In order to implement searching for classes, we plan to adapt our Trie implementation from the autocomplete pset. We will keep a list of class names, and autocomplete class names as they

are searched for by the user. We will also make the autocompleted classes come with a join button, so users can easily join the classes they search for. Search will only result in active classes, as we do not want people to be able to access previous years' resources if they weren't in the class. Thus, classes marked as closed will only be accessible to those who joined before it was closed by clicking on the class name under the 'Taken' list. We have also decided not to implement tab viewing (unlike in this wireframe) for better UI - the app will be like we demoed in the MVP, with classes being clickable to navigate to their pages.