Case Study: HackerNews

March 13, 2024

Abutalib Namazov
abutalib@mit.edu

Abstract

HackerNews[1] is a platform that is unadorned yet quite tailored with distinct rules that foster a focused and engaged tech-centric community. I will design it using concepts and then build it using Kodless without writing any code. Originally, HackerNews is written in Arc Language[2], a dialect of Lisp.

1 Introduction

First, I explain how HackerNews works in detail. We will be building almost all of the features – some are not mentioned in favor of brevity, they are an exercise to the reader.

1.1 Home

If you go to the website directly at https://news.ycombinator.com/, you will see a list of links – *stories*. Each story has a title, an optional link (url), and an optional text, though at least one of url or text needs to be provided. Most stories have a link. For those with links, clicking on the title will take you to the link. In addition, following the story title, you'll see the source of the url. For those without the links, clicking on the title will take you to the story itself. For example, in Figure 1, stories 9 and 18 don't have urls because there is no source.

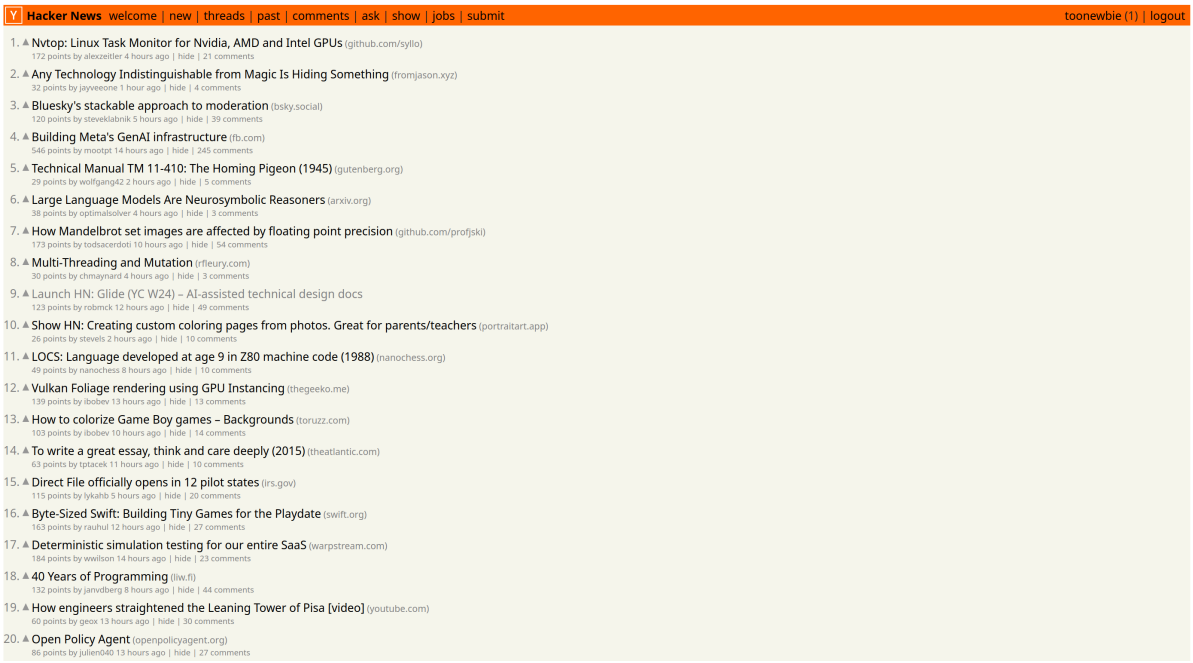

Figure 1: HackerNews home page

In addition, for each story, there are the following elements:

- An up arrow icon – clicking on it will upvote the story;
- The number of points the story has;
- The author of the story – clicking on it will take you to author's profile

[1] https://news.ycombinator.com/
[2] http://arclanguage.org/

- The amount of time passed since the story – clicking on it will take you to the story;
- A hide button – clicking on it will hide the story;
- The number of comments – clicking on it will take you to the story.

These stories, on the main page, are mainly *ranked* with a basic algorithm that divides points to time to a power – the exact form is not public. However, in Arc source code, we can see the formula that is likely to be similar in HackerNews:

$$\text{Score} = \frac{\text{Points} - 1}{(\text{Time in hours since submissions})^{\text{Gravity} = 1.8}}$$

HackerNews also mentions that there are other factors that affects rank such as *user flags* and anti-abuse software.

There's also new page[3] that is the same as home page except that stories are sorted from newest to older.

1.2 Ask

Ask page[4], shown in Figure 2, is stories that ask questions from the community and have no url. The titles of stories usually start with "Ask HN", but moderators sometimes add stories that don't start with "Ask HN" there too. Stories need small amount of points to appear on this page (as small as 2 points). However, all stories without a url appear in asknew page in order of submission: https://news.ycombinator.com/asknew

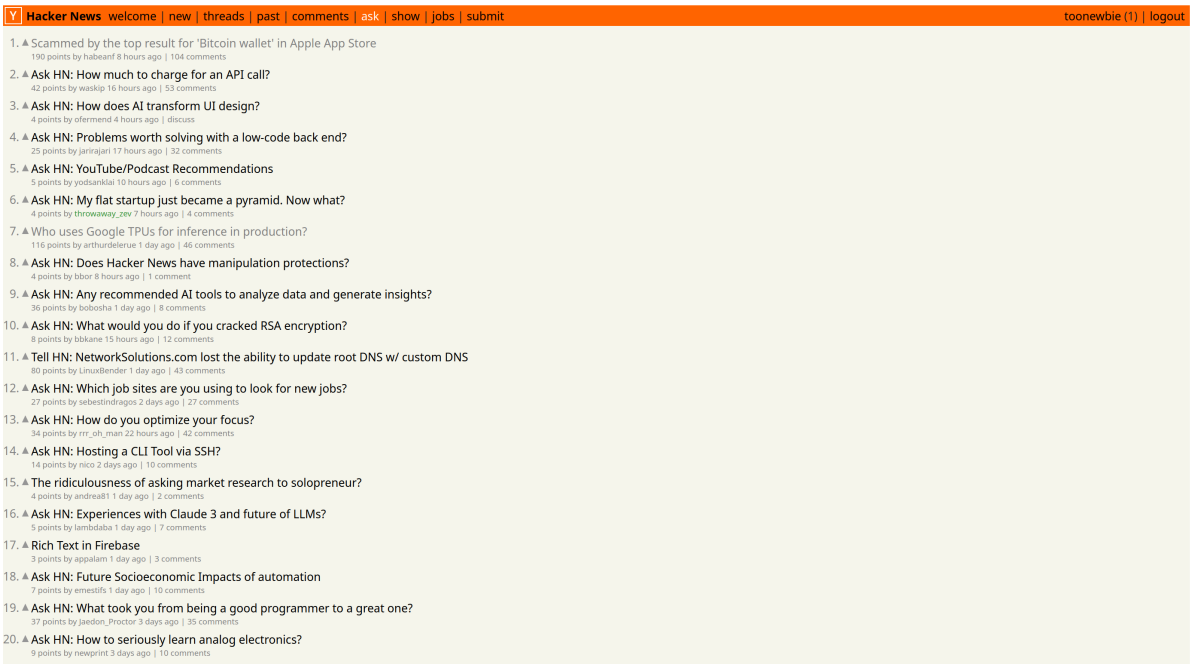

Figure 2: HackerNews ask page

1.3 Show

If the title of the story starts with "Show HN", then it may appear on the show page[5], shown in Figure 3. This page is for stories "for something you've made that other people can play with." The story title must start with "Show HN" for it to appear here given enough points (seemingly, as small as 2 points).

[3] https://news.ycombinator.com/newest
[4] https://news.ycombinator.com/ask
[5] https://news.ycombinator.com/show

Again, all show stories will appear in shownew page in order of submission: https://news.ycombinator.com/shownew

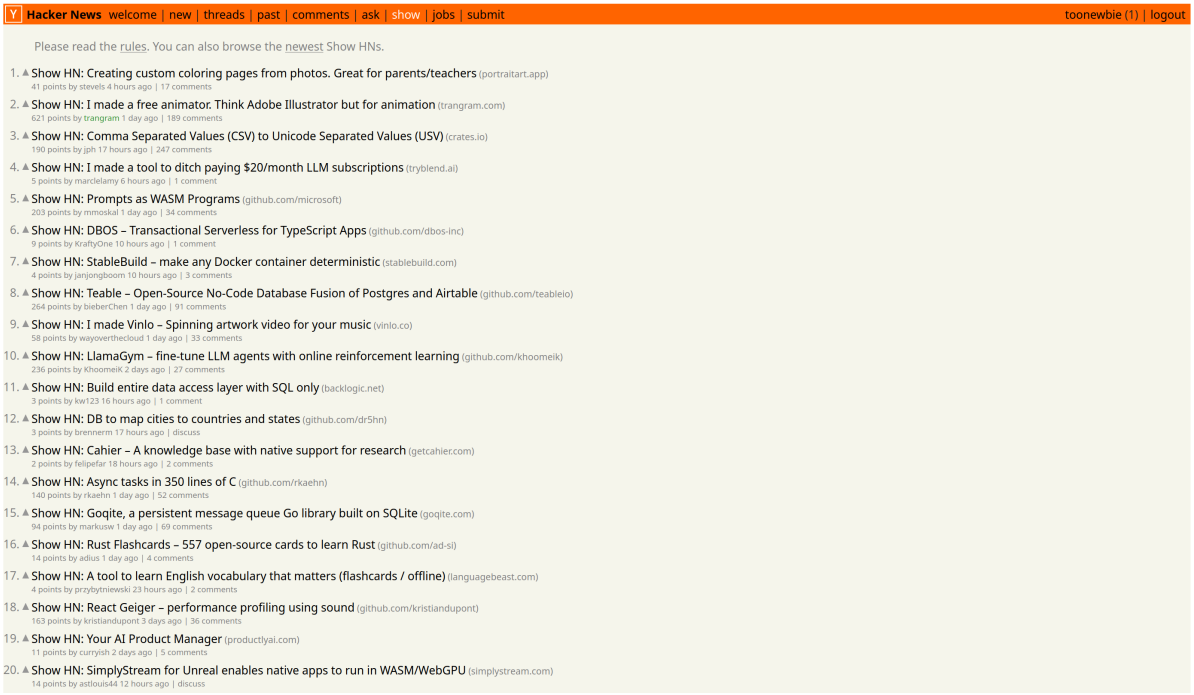

Figure 3: HackerNews show page

1.4 Items

An item refers to either a story or a comment with replies under it, as comments. They can be accessed at link `https://news.ycombinator.com/item?id=item_id` where `item_id` is a unique id for that item. In this sense, both stories and comments are items. See a story in Figure 4 and a comment thread in Figure 5.

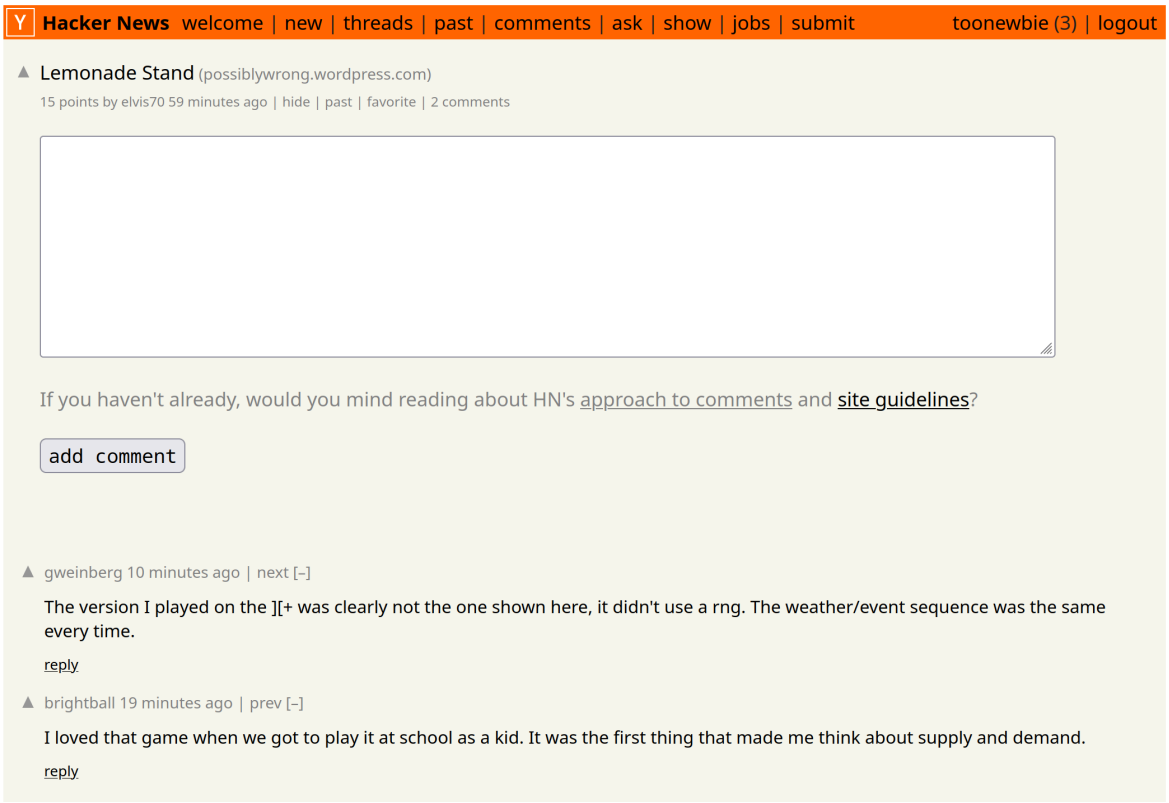

Figure 4: HackerNews story thread (https://news.ycombinator.com/item?id=39694353)

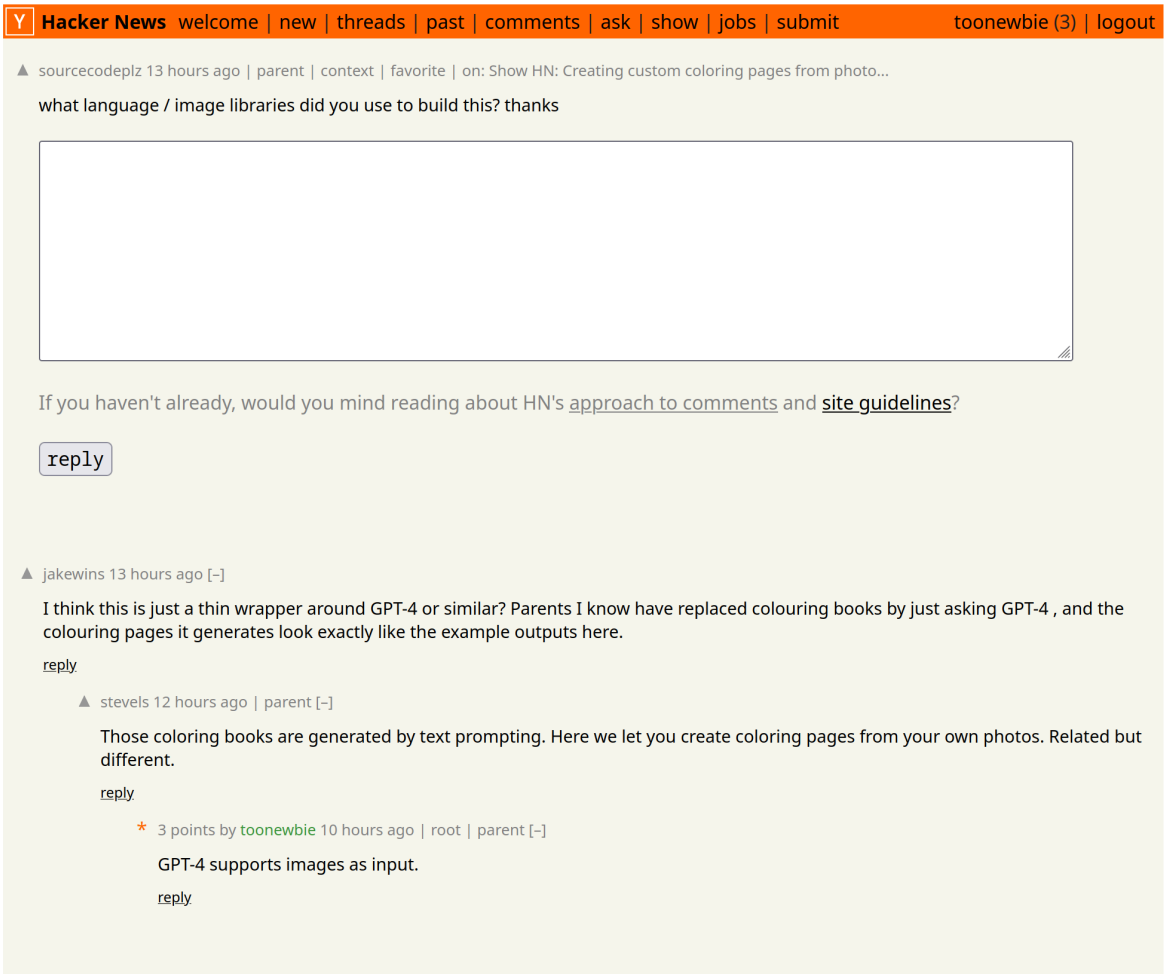

Figure 5: HackerNews story thread (https://news.ycombinator.com/item?id=39688356)

In addition, comment threads will contain links to the *parent* and *context*, referring to, respectively, the direct replied item and the story of this comment is under. It also shows you the context title, looking like with "on: ...", which is also clickable.

It is possible to see your anyone's comments under link `https://news.ycombinator.com/threads?id=<username>` and submissions under link `https://news.ycombinator.com/submissions?id=<username>`.

For comment threads, there is also a minus/plus sign that can be used to collapse the thread for better visibility.

1.5 Karma

A new user starts with karma of 1. Each upvote by another user will increase karma by 1 and each downvote will decrease it by 1.

1.6 Behavioral Details

Thanks to minimaxir for "A List of Hacker News's Undocumented Features and Behaviors"[6] for some details here. A few of the points below quote from this text.

- Items are closed to replies after 2 weeks, or if the submission has been killed by the admins/flagging.
- Comments with negative amount of points will start to fade away, as can be seen in Figure 6.
- New users (account age less than 2 weeks) will have their usernames appear green.

[6]https://github.com/minimaxir/hacker-news-undocumented

- After a submission or a comment is made, it can be edited by the author only within 2 hours. It can also be deleted by the author within those two hours, but only if it has no replies, in order to prevent discussion from being lost.
- Collapsing comments will be remembered for your profile. [flagged] comments are sometimes collapsed by default, and moderators can set a comment to automatically be collapsed if necessary (e.g. meta-discussion).
- After you reach 501 karma, you can downvote comments.
- After you reach 31 karma, you can flag submissions. Although submissions cannot be downvoted, flags act as a "super" downvote and enough flags will strongly reduce the rank of the submission, or kill it entirely (flagging is supposed to be used for submissions which break the site guidelines, but that isn't always the case in practice). A submission that's flagged to death will have a [flagged] tag. Comments behave similarly.
- A [dead] submission (that does not also show [flagged]) is killed by a moderator or by the software. They will only be shown to users who have showdead setting enabled in their profile. A submission can simultaneously be [flagged] and [dead].
- If a user has 31 Karma, they can also vouch for a [dead] submission/comment. A vouched submission/comment has its rank restored (and potentially improved as the vouch can counteract the effects of flags).
- If a user has 251 Karma, they can set the color of the top bar in their profile settings. The default is `#ff6600`. The Y Combinator logo will change its color to match the top bar.
- If the comment depth is 3 or more, reply links are withheld until the comments age a while. The amount of aging is a function of the depth. You can get around it by clicking on the comment's timestamp to go to its own page.

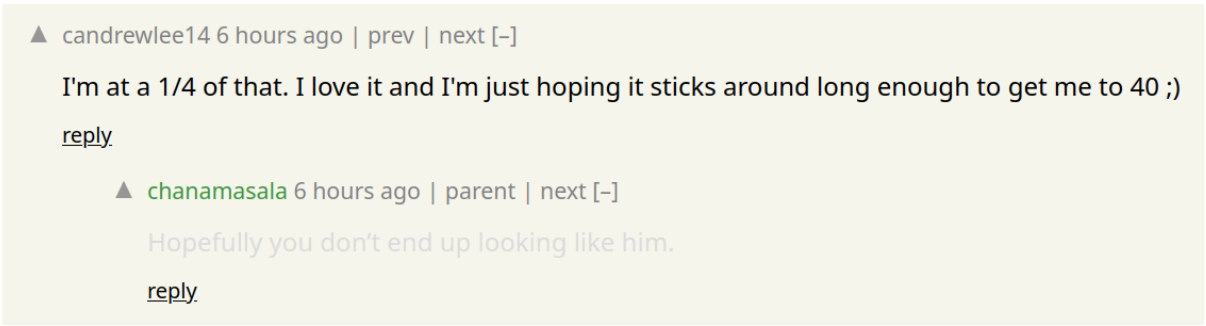

Figure 6: HackerNews comments

2 Recap of Features

We have discussed how HackerNews works. Before going into concept design, we want to write down an informal specification, e.g. planned features for the app. We relax some features for brevity. These changes are underlined.

- User registration and login.
- Users should be able to update their profile: about, email fields. Users also can change their password when logged in.
- Logged in users can submit a story with a title, optional url, and optional text.
- Stories appear on the following pages: `/` (home), `/newest`, `/ask`, `/show`, `/asknew`, `/shownew`. Also at `/front?day=2024-03-12`, which default to yesterday when `day` is not given (UTC).
  - Ask and Show have requirement of respectively "Ask HN" and "Show HN" prefix in the title.
  - Pages show at most 30 stories at a time.
  - Sorted lists show posts at most 4 days old.
- Users can upvote stories and comments which maintains sort order for them.
- Users can add comments to a story or reply to other comments (no 2-week rule).
- Users can favorite and hide stories (no favoriting comments).
- For each upvote by others, user gets 1 karma point. Once reached 6 karma, users can downvote comments. A downvote decrease karma by 1 point.
- Users can see their: comments, hidden, upvoted submissions/comments and favorited submissions. Hiddens and upvotes are not public, but the rest is public (others can see, just like the user).
- Users can change color of their top bar after reaching at least 2 karma.
- Submissions or comments can be edited or deleted within 2 hours of posting. No deleting allowed if there were replies.
- Users with at least 5 karma can flag stories or comments, which will not affect points directly but each flag will act like 5 downvotes in rankings.

Frontend-only features:

- Users can collapse comments.

3 Kodless Prompts

3.1 Concepts

concept: **User**. Each user should have username (string), password (string), optional email (string), optional bio (string), and topBarColor (hex string with default `#ff6600`). Actions should be

- create(username, password)
- update(id, email, bio)
- authenticate
- changePassword
- getById
- getByIds, mapping from id to user
- getByUsername
- changeTopBar

concept: **Post**. Each post should have author (generic), title (string), optional url (string), optional text (string). Actions should be

- create
- update
- delete
- getByAuthor
- get(options?: {dateStart?, dateEnd?, page?, count?, titlePrefix?}) where default page=1, count=30
- getById
- getByIds, mapping from id to post
- isUserAuthor

Update and delete can only happen within 2 hours of creation. All get actions should return sorted results based on creation date.

concept: **Comment**. Each comment should have author (generic), parent (generic), root (generic), and content (string), depth (number). root refers to the root of the comment chain and usually is another generic type. Actions should be

- create(author, content, parent) where parent could be a comment or another generic (in which case it's also the root).
- update
- delete, only works if given comment has no comments under it
- getById
- getByAuthor
- getById
- getByIds, mapping from id to comment
- getByParent(id), which should return comments in the subtree of id (could be comment or other generic) as flattened list
- countByRoots(roots), mapping from each root to their comment count
- isUserAuthor

concept: **Vote**. Each vote should have author (generic), item (generic) and vote type (up or down). Actions should be

- upvote
- downvote
- unvote
- getUpvoted(author)
- getVote(author, item)

- getVotes(author, items?), mapping from item to vote
- getItemPts, should return difference between upvotes and downvotes
- getItemsPts(ids), should return mapping from each id to its point

If user has voted for an item, they must unvote first to vote again.

concept: **Karma**. Each karma should have user (generic) and points (number). Actions should be:
- increase(user, x)
- decrease(user, x)
- get(user),
- isAllowed(user, threshold), throws error if user doesn't have at least threshold

concept: **Mark**. Each mark should have user (generic) and item (generic). Actions should be:
- mark
- unmark
- getByUser
- getByItem
- isMarked, returns if item is marked by user

3.2 App Definition

Instantiate these concepts: Websession, User, Post, Comment, Karma, Vote as PostVotes and CommentVotes, Mark as Favorite, Hide, Flag.

3.3 Syncs/Routes

3.3.1 Basic User routes

- `POST /register` – register and login. new users start with 1 karma.
- `POST /login` – login
- `POST /logout` – logout
- `GET /session` – Return currently logged in user, inside the same object `karma` should be the karma points of the user.
- `GET /users?username=<username>` – Returns the requested user and their karma as `karma` in the same object. If the current logged in user is not this user, hide email and top bar color.
- `PUT /users` – Update bio and email.
- `POST /password` – Change password.

3.3.2 Posts and actions

- `POST /posts` – Create a post and upvote it. Does not affect karma.
- `GET /posts/?count=<count>&page=<page>&prefix=<prefix>&date=<YYYY-MM-DD>` – Get posts sorted by: (p-1)/(hours_since_creation)^1.8-5f formula where `p` is points of this post and `f` is the number of times it got flagged. If date is given, filter by that date. Apply paging after sort. Return posts, and in the same object as each post, `points`, `comments` (number of comments), `author` (user object), `voted` (if the currently logged in user voted; if no logged in it's false), `index` (paged index of the post, starting from 1). Do not include hidden posts. Use default of count 30 and page 1, no prefix. All query params are optional.
- `GET /posts/recent/?count=<count>&page=<page>&prefix=<prefix>` – Get posts by by recent. Return posts, and in the same object as each post, `points`, `comments` (number of comments), `author` (user object), `voted` (if the currently logged in user voted; if no logged in it's false), `index` (paged

index of the post, starting from 1). Do not include hidden posts. Use default of count 30 and page 1, no prefix. All query params are optional.

- `GET /posts/:id` – In the same object, return the post, `points`, `author` (user object), `voted` (if the logged-in user has voted on it, false if no logged in), `hidden`, `favorited`, `flagged`.
- `PUT /posts/:id` – Edit a post.
- `DELETE /posts/:id` – Delete a post, but not allowed if it has comments under it.
- `POST /posts/:id/vote` – Vote the post (only upvote or unvote). Cannot vote self-post. Affect karma of voted user by 1.
- `POST /posts/:id/favorite` – Favorite the post.
- `DELETE /posts/:id/favorite` – Unfavorite the post.
- `POST /posts/:id/hide` – Hide the post.
- `DELETE /posts/:id/hide` – Unhide the post.
- `POST /posts/:id/flag` – Flag a post, requires 5 karma.
- `DELETE /posts/:id/flag` – Unflag a post.
- `GET /favorites?username=<username>` – Get favorited posts of user with given username. Return posts, and in the same object as each post, `points`, `comments` (number of comments), `author` (user object), `voted` (if the currently logged in user voted; if no logged in it's false), `index` (starting from 1).
- `GET /hidden` – Get hidden posts of the logged in user. Return posts, and in the same object as each post, `points`, `comments` (number of comments), `author` (user object), `voted`, `index` (starting from 1).

3.3.3 Comments

- `POST /comments?parent=<id>` – Make a comment by taking in `content` and `parent` and upvote it.
- `GET /comments/:id` – In the same object, return the comment, `author` (user object), and `vote` (if the logged-in user has voted on it, return vote type else null).
- `PUT /comments/:id` – Edit a comment.
- `DELETE /comments/:id` – Delete a comment, but not allowed if it has comments under it.
- `POST /comments/:id/vote` – Vote the comment (upvote, downvote, or unvote). Cannot vote self-comment. Affect karma of voted user by 1. Downvote requires 6 karma.

3.3.4 Misc

- `PUT /topbar` – Change topbar color of the logged in user, requires 2 karma.