6.170 - Project ibetcha
Team: Zulsar Batmunkh, Saadiyah Husnoo,
Dong Hyug Lim, Dana Mukusheva

Design Document

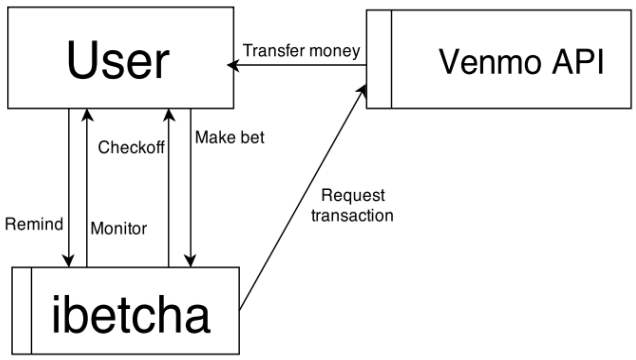
Motivation

ibetcha is an online app to help users follow through with their resolutions by betting money on it with friends. It enables users to build a support network, select a group of trusted friends and seek their support to see to it that he/she completes the task. We think that doing so will add peer pressure and monetary incentive to the user's resolutions. The task can be a one-time task or a recurring one.

Purposes

- **Help a user follow through his/her resolutions.** While it's easy to make resolutions, they're often hard to keep after the first few days. Therefore, ibetcha will help users stick to their resolutions and not to give up even when it gets harder to follow.
- **Make a reliable support network for the user:** ibetcha allows the user to choose a group of trusted friends to monitor him/her. It does so by providing incentive for the support network to regularly check on the user's progress.
- **Help a user add monetary incentive to a resolution:** ibetcha allows a user to increase the stakes associated with the resolution(bet) by allowing him/her to associate a monetary value to the resolution and betting that amount. In addition, the amount of money deters the user from giving up on his/her resolution because the user will lose all the money he initially placed on the bet if he gives up.
- **Help a user add social incentive to a resolution:** When a user completes or fails on a resolution, the friends will know, making peer pressure also an incentive to follow through with the resolution.

Context Diagram



The app is accessible to any user who has a valid Venmo account. We use the Venmo API to proceed with money transfers. The user can invite new users by sending email invitations.

Concepts

ibetcha incorporates the following concepts to support our purposes:

Betcher: a user who makes the bet, puts money on a stake and chooses his/her monitors in order to fulfill his/her resolution.

Bet: the representation of the deal between a betcher and his/her monitors. The Bet contains such fields as a Bounty (see below), Monitors (see below), a creator, frequency (e.g., daily, weekly, weekends only, etc.), start date, end date. The Bet is a central concept of our app and fulfills the primary purpose of helping users to follow their resolutions.

Monitor: a friend chosen by the user to keep track of the user's progress. The user has a group of three or five monitors for progress. The monitor is empowered to give Checkoffs (see below) to the betcher.

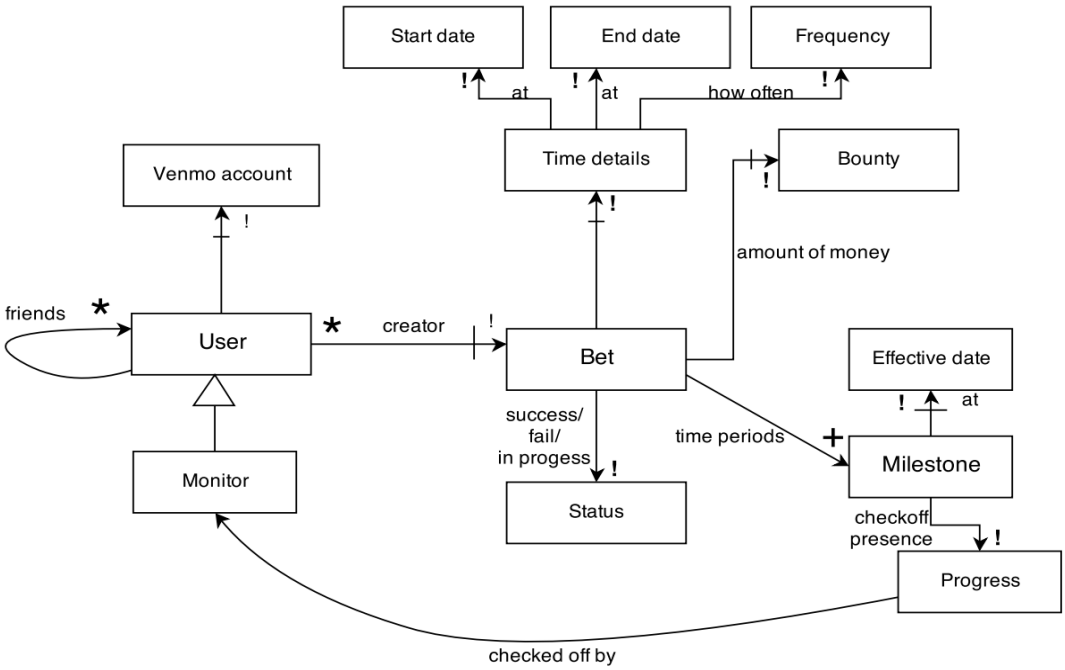
Bounty: amount of money put on the bet. The bounty is chosen by the betcher and can't be changed after the bet was created. In case of successful task completion, the bounty is returned to the betcher, otherwise, it is evenly distributed between monitors. The bounty is used in the user/system interactions and system/Venmo interactions.

Checkoff: a confirmation of the completion of the entire task or the part of the task. Only the monitors can give a checkoff to the betcher

Milestone: a state that represents the checkoff received for a specific point in time. A milestone is "success" or "failure" depending on the checkoff, or it can be "pending" if the checkoff has not been received for that time.

The betcher is the user

Data model



The above diagram demonstrates our choice for data representation. The main components are the User, the Bet and the Milestone. For each Bet, a user can either be a Monitor or the Creator.

The central concept of the app is represented as the Bet object. A Bet can have only one creator and a fixed number of Monitors. A Bet has attributes such as a list of Milestones, timing details (immutable), creator (immutable), list of Monitors (immutable), Bounty (immutable), and Status. Possible statuses are “Open”, “Closed”, “Action Required”, “Dropped”. The Bet object references multiple Milestone objects. Each Milestone object represents a single checkpoint for the bet. Each Milestone object has an effective day and the indicator of the progress (possible values include “Pending Action”, “Open”, “Closed”, “Inactive”). Only a Monitor can mutate Milestone progress. A User object must have an associated Venmo account information.

Behavior

- **User Interface workflow**
  The workflow of the app includes such actions as:
  - a User signs up for an iBetcha account and adds Venmo account
  - a User creates a Bet object, chooses the Bounty and Monitors (at least 3 no more than 5) among his friends, chooses the timing details of the Bet. User cannot edit or delete the Bet.

- friends selection process: a User can invite his/her friends (handled via email) to be his/her ibetcha friends (or create an ibetcha profile and then become a friend).
- monitor selection process: a User can select Monitors for his/her new bet from the list of his/her ibetcha friends.
- a User can be invited to be the monitor of the Bet and checkoff the Milestones of the Bet creator.
- a User can reject an invitation to be a Monitor.
- a User can review the profiles of other Users to check their resolution statistics.
- a User who is a Monitor to some Bet will receive a reminder if none of the fellow Monitors of the same bet did not check the Milestone after its effective date has passed.
- if a Monitor decides to fail the Milestone, he/she will be asked to confirm his/her actions and possibly, leave some feedback.
- a Bet creator will be able to see the final result of his/her bet after the end date has passed.
- Monitors will receive money in case the Bet they were monitoring has failed.
- if all Monitors do not respond to the emails and do not checkoff the user, when the End date passes the Bet creator will NOT lose the money.

- **Security Concerns**
  - Since money is involved with ibetcha, there's a lot of risk involved regarding user protection. Extensive measures should be taken to protect users from unauthorized charges, credit card frauds or theft of personal information or other crimes.
  - Not having robust security system, thereby exposing our website to cross-site scripting attacks. As a result, attackers could inject scripts into our app and users could be redirected to web content controlled by the attacker under the guise of our app.
  - Not correctly implementing authentication and session management might allow outside attackers to compromise users' passwords or personal information and assume users' identities.
  - Hackers might get into our system disguised as authenticated users and might disrupt the app, interacting with other users and falsely checkoffing them to hoard their money.
  - Malicious users might try to drown our server with a vast number of requests or actions that will degrade our app performance (could create multiple number of bets with maximum frequencies, and creating many pending checkoffs, that would require a lot of memory repeatedly and sending repeated emails to users).
  - When users invite other users via email, we need to make sure that no other information is exposed, and the url included in the email is not used for purposes other than signup or invitation

- We need to make sure that a user cannot spam people via our email functionality.

- **Mitigations:**
    - To secure users personal information further, we use Passport.js to authenticate the users and hash passwords while stored in the database. Moreover, we'll minimize the sensitive data stored in ibetcha by using Venmo to secure money transfers. Therefore, all transactions will be processed through Venmo and users' credit card and other personal information would not be stored in ibetcha.
    - We ensure that monitors can be chosen only from the friends list; thus no strangers can modify user's profile and his/her ongoing bets. We also encourage interaction between users as an informal identification to identify outside attackers as early as possible. Moreover, we could also introduce extra security questions whose answers are only known to the trusted parties to secure user identities.
    - Furthermore, most of the objects created in the back-end are immutable and safely stored in the MongoDB. Only recognized and authorized users can access and mutate fields. We will also sanitize our inputs.
    - For all API requests, we can check for the login information and return false if not logged in.
    - In all emails, we do not expose any personal information. In addition, we can check if a user has an email before sending an email, or we can make the email a required field for signup process.
    - To prevent spams, we can limit emailing non-users to just invites. The rest of the emails, we check if the recipient of the emails are the members of ibetcha. Also, the user does not get to write the content of the emails.

Design Challenges

- **How to check-off the user and make sure that the bet is completed?** How to track the user progress if the bet is expected to be over a long period?

  Potential Solutions:

    - *User will select as many monitors as he desired, who will track his/her progress.* With this option, the checkoff system would be more flexible and would not depend on the monitors too much. Nevertheless, having many monitors delegates the responsibilities of the monitors, and more monitors would be likely to depend on each other and not do their checkoffs. If there are too many monitors, it would complicate the checkoff system and furthermore, we would need to have a universal checkoff strategy that is applicable for all different possibilities.

- *User will selected either 1 or 2 trusted monitors.* Having either one or two monitors will have the advantage of having trusted monitors who would be likely to do their checkoffs on time. Nevertheless, the checkoff system will depend too heavily on the responsible monitors (if they're busy) and also if the number of the monitors is too few, the bet would lack peer pressure.

- *User should be able to select only a limited number of monitors (3 to 5).* Having not too much, but not too few monitors, provides nice balance between dependence on monitors and also the checkoff strategy. If one monitor is busy, others could fill in his position by checking off the user. Furthermore, in this case, since the user's selecting his/her trusted friends, maybe only one checkoff would be sufficient to confirm user's progress.

- *User will have the option to select a different frequency for the bet and his/her progress will be tracked over milestones.* If the bet is over a long time, this provides nice flexibility for the track of the user progress and keeps the user motivated over the long-term. Furthermore, user would be able to see his/her progress over different periods.

Our choice:

- Therefore, we decided to proceed with the option of choosing only a limited number of monitors (currently, we're deciding the number to be in range from 3 to 5) with also the option to have a different frequencies for the bet. As noted above, this would make the checkoff system simple and consistent and also decentralize the responsibilities between monitors.

- **How to prevent users from betting trivial amounts of money?**

Potential solutions:

- *The User will have a minimum amount of bounty(threshold) required per bet.* The bounty threshold should not be either too much or too small.

- *We assume that user will bet however much his resolution is worth.* Peer pressure will check back on this; Bob's friends would judge him if he bets 10 cents on his resolution to get a beach body.

Our choice:

- Currently, we're deciding the range of the bounty to be -> possibly minimum -> $5 and maximum -> $50.

- **How many checkoffs are needed to confirm that the user succeeded?** What if there are inconsistencies in the checkoffs?

  Potential Solutions:

  - *Taking the majority of the checkoffs* -- If there are inconsistencies among the checkoffs, this would be a nice way to resolve the conflict. However, this would be difficult to coordinate among multiple monitors since one or more monitors may be out of reach (no Wifi, psets, vacationing in Hawaii… etc)

  - *Only one checkoff is enough (either failed or succeeded)* -- Since user is selecting his/her trusted friends, it will not be very likely that the friends would be interested in falsely reporting user's progress and claim that the user failed when in fact the user succeeded his/her goal. Furthermore, there would be a limited number of monitors (either 3 or 5), thus only one checkoff would be sufficient. However, this approach faces the problem of "delegation of responsibility." Each person in the group may think "Well, another monitor can do the check-off."

  Our choice:

  - Choose the second option and require only one checkoff. If during either one of the frequencies, if one of the monitors say that the user did not follow his resolution, then the bet is over and user will be considered as failed at his/her bet. Similarly, if one monitors say that user actually did what he promised to do, the user will be checked off as having completed his/her duty.

- **How to prevent monitors from falsely checking off the user and hoarding his/her money?**

  Potential Solutions:

  - *After the decision is made, ask other monitors for confirmation.* It would prevent any monitor from falsely checking off since others also have to agree to the decision. It would be highly unlikely that all the monitors would collude. But this option would be hard to manage since some monitors might be busy or what if all of them are busy? Then, we might need have to have another strategy for concluding other users' confirmation period and acceptance cutoff.

  - *User will select his/her trusted friends as monitors.* Therefore, this will ensure that the monitors will be trustees who have genuine interest in helping their friend achieve his/her goal.

Our choice:

- We'll rely on the user's (the user who made the bet) judgement on choosing his/her own monitors and therefore, chose to follow the second option. If the user has no ibetcha friends, he/she would also have the option to invite his/her friends into ibetcha by sending them emails with appropriate ibetcha signup link.

Team Work Plan

Stakeholders

Our app is geared towards users of any age or group. All that is needed is a Venmo account to process transactions.

Our app has a potential to become an additional service of Venmo, since it will attract more customers to use their system in the everyday life. Thus, Venmo Developers division is a potential stakeholder. Also, the app can be redesigned to be a Facebook application and work within Facebook, which will ease the "friending", "reminding" and "inviting" features.

The admins of the website are the developers of the app, and a general user does not have an access to make changes in any data except for the specified cases (checking off).

Resources

Our app is designed to interact with Venmo. However, since Venmo handles money transaction, the app needs to be approved by the Venmo Developers division in order to access and use their API.

There might also be legal implications because of the nature of our app, which involves betting (sports betting is illegal).

Security enforcement is also a major aspect, given that our app deals with monetary transactions.

Tasks

- ~~Research on external libraries~~
- ~~Implement authentication (passport.js - user signup/login)~~
- ~~Make bets feature (specify content, times, frequency, and monitors, etc)~~
- ~~Add test suites as the we work~~
- ~~Decide on API to be implemented~~
- ~~Document design decisions~~
- Use Venmo API *(after MVP stage)*
- Make "friending" and "invites" feature
- Handle reminders of the pending bets
- Implement automate emailing/updating (using cron job)
- Design client side code (using Angular.js)

- Design User Interface *(after MVP stage)*

Minimum Viable Product

Our plan for MVP stage of the development is to finish the main functionality of the app, including user signup and authentication with passport.js, creating bets, inviting friends via email, remind the monitors about pending checkoffs and notifying involved users about the outcome of the bet. Users will be able to view profiles of other users to see the statistics of the bet history.

On this stage we postpone interactions with Venmo API because we need to get approval to use their API. Therefore, the actual money transfers and adding Venmo account are not functional. Instead, we will implement a system to track the amounts owed as well as to whom the amount is owed(payee). The payee can then "clear dept" to mark that transaction as completed. In this manner, we can simulate the money transfer mechanism without involving actual money.

Also, until we resolve all the issues with APIs, we do not proceed with the design of User Interface.

Tentative API Design:

USER:

METHOD	URL	Description	JSON returned
POST	/users/login	login to ibetcha	{success: true, content:user JSON} {success:false, err: error}
GET	/users	get a list of users	{success: true, content:JSON of users} {success:false, err: error}
POST	/users/	signup a new user	{success: true, content:User JSON} {success:false, err: error}
GET	/users/:user_id	get user profile	{success: true, content: User JSON} {success:false, err: error}
PUT	/users/:user_id	edit user	{success: true, content: empty} {success:false, err: error}
GET	/users/:user_id /bets	get all bets associated with a user	{success: true, content: [Bet Objects]} {success: false, err: error}

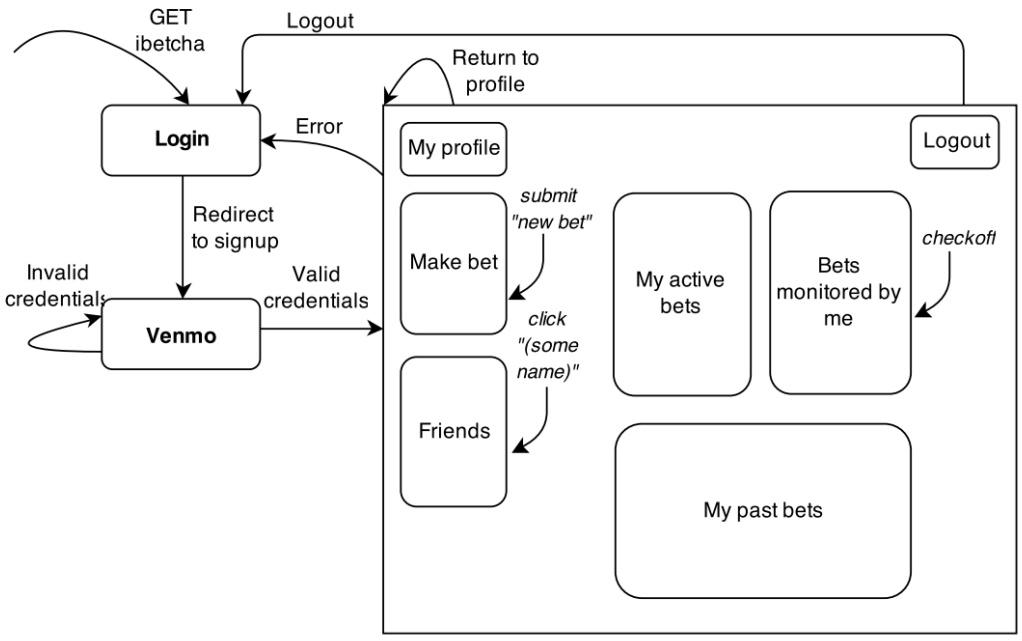
BET:

METHOD	URL	Description	JSON returned
POST	/bets	create a Bet object	{success: true, content: Bet object} {success:false, err: error}
PUT	/bets/:bet_id	edit a Bet object	{success: true, content:Bet object} {success:false, err: error}
GET	/bets/:bet_id	get a Bet object	{success: true, content:Bet object} {success:false, err: error}

Milestones:

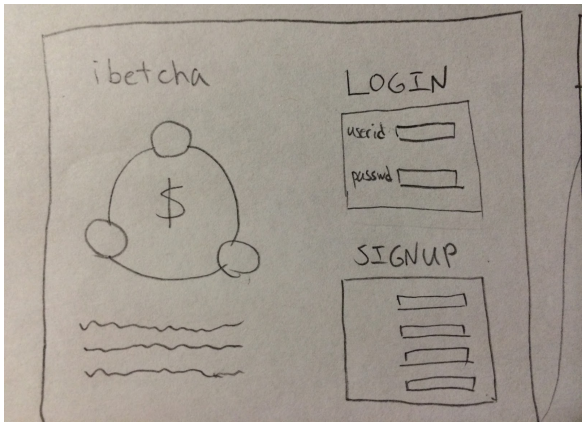
METHOD	URL	Description	JSON returned
PUT	milestones/milestone_id	edit a Milestone object	{success: true, content: Milestone object} {success:false, err: error}

Tentative Main Page flow

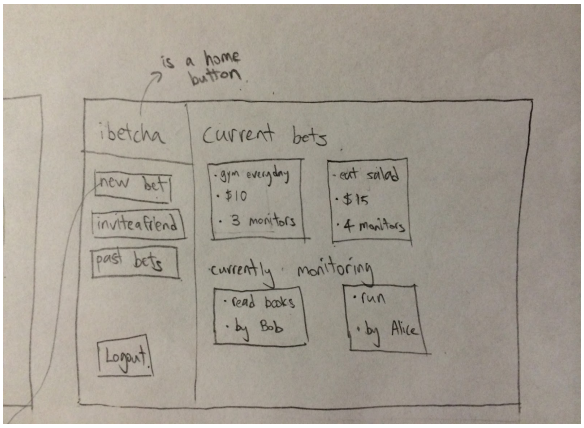


Wireframes:

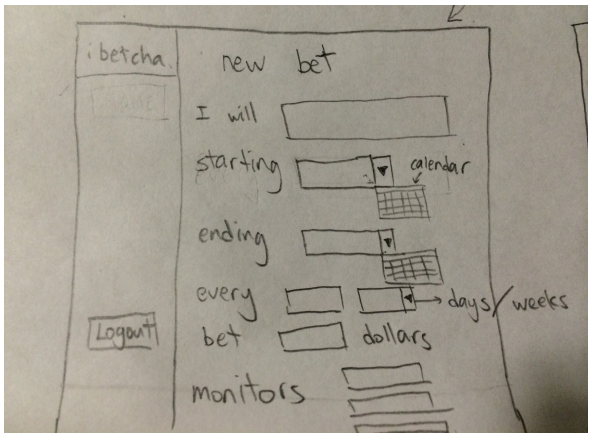
login/sign_up page page



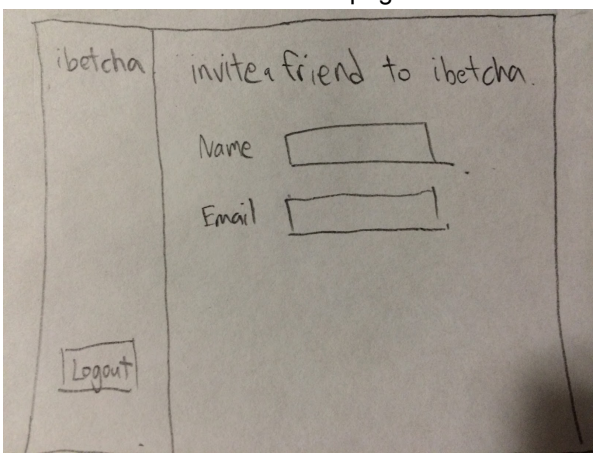
Overview of bets



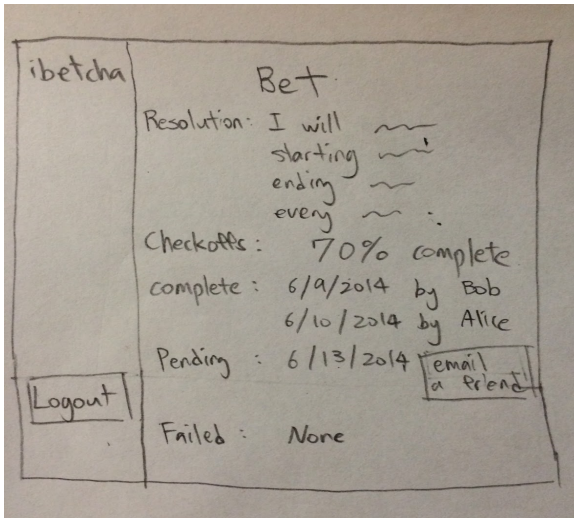
Bet posting form



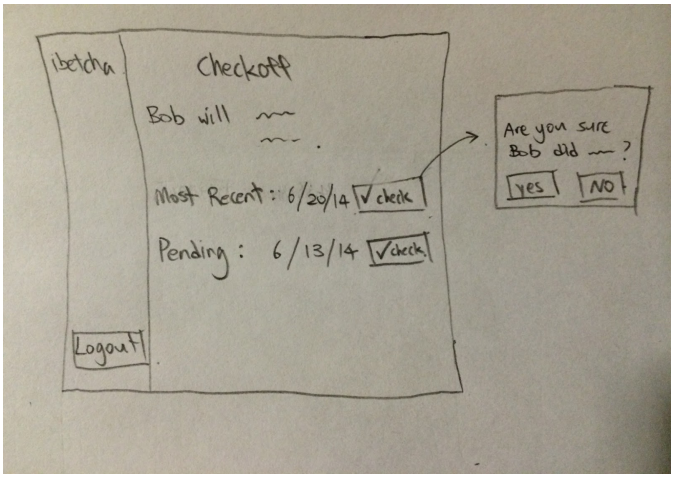
Friend page



Bet overview Page



Bet checkoff page



Note: The side box represents a confirmation dialog box