Thursday, November 20, 2014

Progress Report:
This week we implemented our MVP. We started by building our REST API, which required some changes to our data design. Specifically, items, offers, and users needed to know more about each other than we previously thought, in order to perform actions such as offer matching, etc. One challenge we encountered was “deep population” of fields. Oftentimes, we would want to retrieve some data such as the reviews of all transactions involving a particular user. However, populating several levels down is difficult using Mongoose, and we ran into issues of circular dependencies.

We tested our API using Postman. We also added items using Postman, since the app currently does not allow users to add items.
Our app only allows users with @mit.edu, @csail.mit.edu and @media.mit.edu email addresses to create accounts. This is done using a simple REGEX check, rather than an email validation library as discussed in the previous meeting.

We create the frontend of our app and used Angular.js to connect it with the backend and populate the page with data from our database. This made us realize that we have to populate many levels of the data in order to have access to them to display it in the views.

Agenda:
- What should we do for security?
- Have to refresh page to reflect updated changes after deleting offer and posting a review
- Try using the page and see if they have any suggestions.
- Should we have a page to view other users’ reputations? (We currently do not allow users to cancel transactions)
- Should users be allowed to add items? (We currently do not have item delete implemented)
- Are there any additional features we should add?
- Do we need to have unit tests? Should we clear the database every time we run the tests?

Outcomes/Decisions:
-