CS252 Course Presentation

SHARE@IITK

Group 1729

November 10, 2016

THE NEED

Every year about 1200 students join the campus community and an equal number leave it. There is a need to pass on things like books, mattresses, coolers, study material and what not. At the end of each semester, students want to give away their books to their juniors and there are many who want to take it.
Our application is aimed at promoting **convenient** *sharing of items.*

Technology Used

We have used **Mongoose with NodeJS** for our application.

- Schemas help in structuring the documents. Mongoose is a rich data modeling library also implements validation to make sure your schema is well satisfied when inserting/updating/finding documents from collections.
- A relational database structures data into tables and rows, while MongoDB structures data into collections of JSON documents. Originally designed for lightweight exchanges between browser and server, it has become widely accepted for many types of applications.
- It as well creates Model abstraction which makes it easier to work with, so it looks like you are working with just objects rather than pure data.
- Since forms don't permit PUT and DELETE methods, the presence of inbuilt functions like findByID, Remove, Update in Mongoose makes implementing CRUD simpler.

User Schema

User Schema:

- local name
- local email
- local password
- List of Notifications
- List of items owned by the user

Item Schema and the need of an Inverse mapping

There is a need of user to item as well as item to user mapping because traversing items through users in mongoose would have been possible only by iterating over all possible values which is highly inefficient w.r.t. to findbyID in Mongoose.
Item Schema:

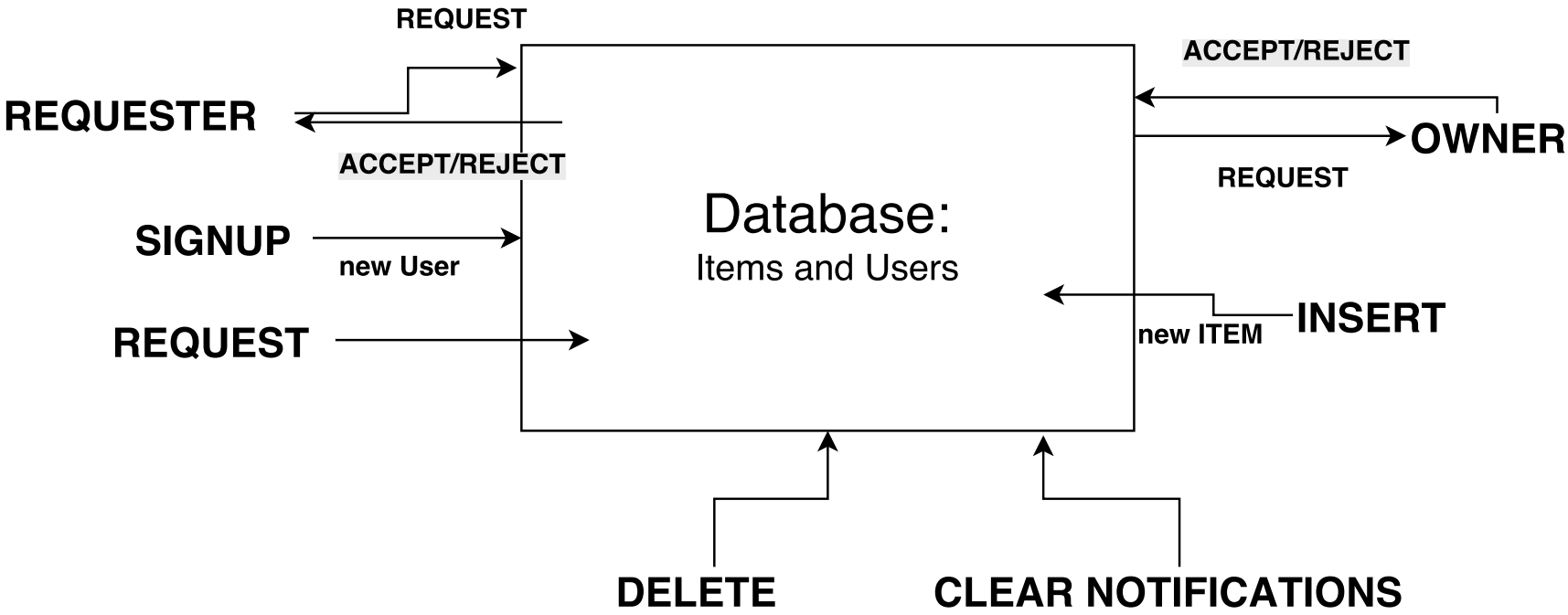
- Name of the Item
- Owner's Name
- Category
- Requester's ID if not null
- Requester's Name
- Item's Status: 0 for deleted, 1 for invisible and 2 for visible item
- Description

Functionalities Implemented

Login: Login using email and password has been implemented.
Search: Keyword matching based case insensitive search on description and item name has been implemented. Check has been applied not to render items pending on request, deleted items or items owned by the user himself.
Request: User has the option to request for an item. The owner of the item is notified of this request.
Insert: User can also insert an item to be shared in the database.
Update Profile: User has the option to update his/her name.

Functionalities Implemented

Delete Item: User can delete an item previously inserted by him for sharing.
Reject Request: User can reject a request for an item. The item becomes visible to all after this.
Accept Request: User can accept a request for an item. The item becomes deleted.
Notify on rejection and Acceptance: In both cases, acceptance and rejection the requester is notified accordingly.
Clear Notifications: User can clear all the notifications.


REQUEST
REQUESTER
ACCEPT/REJECT
ACCEPT/REJECT
OWNER
REQUEST
SIGNUP
new User
Database:
Items and Users
REQUEST
new ITEM
INSERT
DELETE
CLEAR NOTIFICATIONS

Future Work

- Presently, only one user can request for an item. A queue implementation is possible to maintain a list of all the people who have requested for the item.
- A chat option between owner and requester of the item is possible.
- Google Login: We had partially implemented it for just log in but using the information throughout had led to many errors. So we chose to go with the local log in for this project.
- Category Based Search
- We can extend SHARE@IITK to SHAREBUYSELL@IITK.

Timeline of Work

The course project was implemented over a span of three weeks with work equally distributed.

- Initially, since none of us previously acquainted with any server based application framework reading up about all this and figuring out the right database and language took our first week.
- During the next week, we used Passport.js to set up our application's pages, log in and insert option.
- In the last week, we implemented delete, reject, accept, notify and request functions.

Acknowledgment

We would like to thank **Professor Satyadev Nandakumar** and **Professor Piyush P. Kurur** for mentoring us throughout this course.
We would also like to thank **Tejas Gandhi** and **Debjeet Majumdar** for helping us with all our doubts (big or small) and giving regular feedbacks to improve our application.